

AD-A182 874

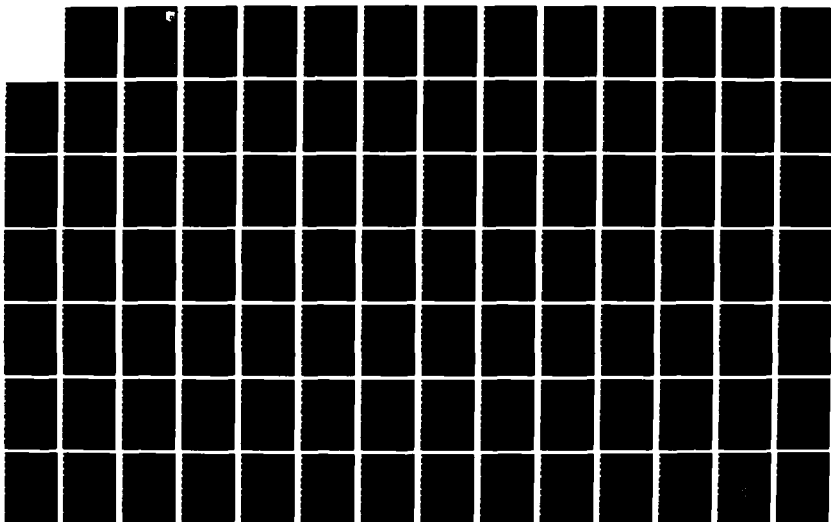
OPTICAL ARCHITECTURES FOR MATRIX PROCESSING(U)
WESTINGHOUSE RESEARCH AND DEVELOPMENT CENTER PITTSBURGH
PA A P GOUTZOULIS ET AL NOV 86 AFMAL-TR-86-1117
F33615-84-C-1499

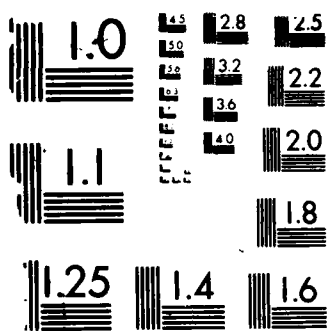
1/2

UNCLASSIFIED

F/G 12/6

NL





MICROCOPY RESOLUTION TEST CHART
 NATIONAL BUREAU OF STANDARDS-1963-A

DWG FILE COPY

AD-A182 874

2



AFWAL-TR-86-1117

OPTICAL ARCHITECTURES FOR MATRIX PROCESSING

A.P. GOUTZOULIS
D.K. DAVIES

Westinghouse R&D Center
1310 Beulah Road
Pittsburgh, Pennsylvania 15235

November 1986

Final Report for Period September 1984 - April 1986

Approved for public release; distribution unlimited.

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

DTIC
ELECTE
AUG 03 1987
S E D

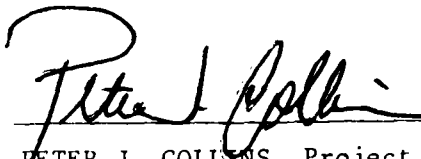
87 7 31 058

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

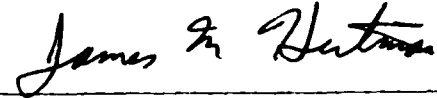
This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



PETER J. COLLINS, Project Engineer
Electro-Optic Techniques Group
Electro-Optics Technology Branch

FOR THE COMMANDER



JAMES M. HEITMAN, Chief
Electro-Optic Techniques Group
Electro-Optics Technology Branch



RICHARD L. REMSKI, Chief
Electro-Optics Technology Branch
Avionics Laboratory

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AADO, W-PAFB, OH 45433 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			CONTINUED ON BACK	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-1117	
6a. NAME OF PERFORMING ORGANIZATION WESTINGHOUSE R&D CENTER	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AVIONICS LABORATORY (AFWAL/AADO-2) AIR FORCE WRIGHT AERONAUTICAL LABORATORIES		
6c. ADDRESS (City, State and ZIP Code) PITTSBURGH, PA 15235		7b. ADDRESS (City, State and ZIP Code) WRIGHT-PATTERSON AFB, OH 45433-6543		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFWAL/AVIONICS LABORATORY	8b. OFFICE SYMBOL (If applicable) AFWAL/AADO-2	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-84-C-1499		
8c. ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6543		10. SOURCE OF FUNDING NOS.		
		PROGRAM ELEMENT NO. 62204F	PROJECT NO. 2001	TASK NO. 02
				WORK UNIT NO. 95
11. TITLE (Include Security Classification) OPTICAL ARCHITECTURES FOR MATRIX PROCESSING				
12. PERSONAL AUTHOR(S) GOUTZOULIS, A. P. AND DAVIES, D. K.				
13a. TYPE OF REPORT FINAL	13b. TIME COVERED FROM 9/3/84 TO 4/15/86	14. DATE OF REPORT (Yr. Mo. Day) NOVEMBER 1986	15. PAGE COUNT 187	
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary; and identify by block number)	
FIELD	GROUP	SUB. GR.		
20	6		Optical processing; optical computing architectures; acousto-optic processors; residue arithmetic; look-up table E-0	
17	9	7	processing; Digital multiplication by analog convolution	
19. ABSTRACT (Continue on reverse if necessary; and identify by block number)			(OVER)	
<p>A study to explore optical methods for performing eigen system calculation based on matrix-vector or matrix-matrix multiplications has been completed. Several implementations of eigen system solution algorithms including digital multiplication by analog convolution and bit parallel multiplication were analyzed to determine performance/cost tradeoffs. All methods were determined to have unsatisfactory multiplication speed and efficiency. Two new approaches to the problem, optically interconnected electronic multipliers and position coded residue opto-electronic look-up table (LUT) processing, were proposed. After analysis and LUT implementation, both methods appear feasible and merit further investigation.</p>				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL PETER J. COLLINS			22b. TELEPHONE NUMBER (Include Area Code) (513) 255-5582	22c. OFFICE SYMBOL AFWAL/AADO-2

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

BLOCK #3.

Approved for public release; distribution is unlimited.

BLOCK #18. (continued)

parallel multiplication, circularly polarizing sampling.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input checked="" type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. INTRODUCTION AND SUMMARY	1-1
2. EIGENSYSTEM SOLUTION	2-1
2.1 Introduction	2-1
2.2 Gershgorin Method	2-4
2.3 Power Method	2-8
2.4 Inverse Iteration	2-9
2.5 Q-R Method	2-12
2.6 Discussion	2-16
3. ALGORITHMS FOR HIGH-ACCURACY ACOUSTO-OPTIC PROCESSORS.....	3-1
3.1 Digital Multiplication via Analog Convolution (DMAC) ..	3-1
3.2 Bit Parallel Multiplication (BPAM)	3-7
4. OPTICAL ARCHITECTURES FOR DMAC AND BPAM.....	4-1
4.1 DMAC Acousto-Optic Space-Integrating Processor	4-1
4.2 DMAC Acousto-Optic Space-Integrating Processor Characteristics	4-6
4.3 DMAC Acousto-Optic Time-Integrating Processor	4-9
4.4 DMAC Acousto-Optic Time-Integrating Processor Characteristics	4-14
4.5 BPAM Acousto-Optic Processor	4-17
4.6 BPAM Acousto-Optic Processor Characteristics	4-21
4.7 BPAM Acousto-Optic Response Analysis	4-22
5. CIRCULARLY POLARIZING SAMPLING TECHNIQUE FOR COMPLEX MATRIX OPERATION.....	5-1
5.1 Circularly Polarizing (CP) Sampling	5-4
5.2 Matrix Multiplication Using CP Sampled Data	5-6
6. PERFORMANCE OF HIGH ACCURACY ACOUSTO-OPTIC PROCESSORS	6-1
6.1 Performance of Electronic Multipliers	6-1
6.2 Performance of DMAC Based AO Processors	6-2
6.3 Performance of BPAM Based AO Processors	6-7
6.4 Performance Comparison Conclusions	6-12

TABLE OF CONTENTS (Cont'd.)

<u>Section</u>	<u>Page</u>
7. OPTICALLY ADDRESSED ELECTRONIC DIGITAL MULTIPLIERS	7-1
7.1 Introduction	7-1
7.2 Optically Interconnected Array Processor for Matrix Multiplication	7-2
7.3 Optical Interconnects for Array Processor	7-4
7.4 Prototype Optically Addressed ECL Multiplier	7-14
8. RESIDUE LOOK-UP TABLE ELECTRO-OPTIC PROCESSING	8-1
8.1 Introduction	8-1
8.2 Residue Arithmetic Basics	8-2
8.3 Residue Look-up Table Processing	8-4
8.4 LUT Experimental Results	8-12
8.5 Binary-to-Residue Conversion (B/R)	8-22
8.6 Residue-to-Binary Conversion (R/B)	8-26
8.7 Hardware Minimisation	8-32
8.8 System Characteristics for a Square Systolic Residue System	8-37
9. RESIDUE LUT IMPLEMENTATION OF GRAM-SCHMIDT APPROACH FOR THE SOLUTION OF LINEAR SYSTEMS	9-1
9.1 Introduction	9-1
9.2 Residue Resolutions of Linear Systems	9-1
9.3 Gram-Schmidt Variant	9-3
9.4 LUT Implementation of Gram-Schmidt Approach	9-9
9.5 Processor Characteristics	9-23
10. CONCLUSIONS AND RECOMMENDATIONS	10-1
11. ACKNOWLEDGEMENTS	11-1
12. REFERENCES	Ref-1
13. APPENDIX A. AN ANALYSIS FOR CIRCULARLY POLARIZED SAMPLING..	A-1

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
2.1	Weight Determination for APAR by Complete Eigenanalysis of the Data Covariance Matrix	2-3
2.2	Flow Chart for the Determination of Eigenvalue Regions using the Gershgorin Method	2-5
2.3	Flow Chart for the Determination of Eigenvalues and Eigenvectors Using the Power Method	2-7
2.4	Flow Chart for the Determination of Eigenvectors Corresponding to Given Eigenvalues by the Method of Inverse Iteration	2-10
2.5	Flow Chart for the LU Triangular Decomposition of a Matrix	2-11
2.6	Flow Chart for Implementing the QR Algorithm	2-13
2.7	Flow Chart for the QR Factorization of a Matrix	2-14
3.1	Example of Multiplication in Twos Complement Representation	3-4
3.2	Example of Conversion from Mixed Binary Representation to Twos Complement Representation	3-5
4.1	Acousto-Optic Processor for Binary Number Multiplication	4-2
4.2	Space-Integrating Acousto-Optic Processor for Accurate Vector-Matrix Multiplication	4-4
4.3	Time-Integrating Acousto-Optic processor.	4-10
4.4	Input Timing Diagram for the Processor of Figure 4.3	4-11
4.5	Time-Integrating Acousto-Optic Processor for Inner Product Formation	4-13
4.6	Time-Integrating Acousto-Optic Binary Systolic Processor	4-15
4.7	BPAM Implementation with AO Cell and λ -Multiplexing	4-18

LIST OF FIGURES (Cont'd.)

<u>Figure</u>		<u>Page</u>
4.8	Product Summation via Detector Area Shape	4-20
5.1	Multiplication of Complex-Valued Matrices Using Real-Valued Matrix Multiplier	5-2
5.2	Complex-Valued Multiplier Cell	5-3
5.3	Circularly Polarizing Sampling Scheme	5-5
5.4	Correlation Operation Using CP Sampling Approach	5-8
5.5	Positional Quadrature Representation in Matrix-Vector Multiplication	5-10
6.1	Counter Arrangement for a 2-bit Multiplier	6-10
6.2	Laser-Diode/AND-Gate System for Product Formation	6-11
7.1	Square Array Processor for Matrix-Matrix Multiplication	7-3
7.2	ML 4402 Laser Diode Pigtail Adapter	7-6
7.3	RFC Splitter Approach	7-8
7.4	Output Radiation Pattern from 1:M RFC Splitters with $M = 7$ and 19	7-12
7.5	Schematic Diagram of the Optically Addressed Multiplier	7-15
7.6	4x4 Bit ECL Multiplier Design	7-16
7.7	Schematic Diagram of the Pesaris Multiplier Arrangement	7-18
7.8	Logic Diagram of the MC 10287 Array Multiplier Block	7-19
7.9	Photograph of the Assembled 4x4 Bit Multiplier Board with Optical Interconnects from the Data Generator Board	7-21

LIST OF FIGURES (Cont'd.)

<u>Figure</u>		<u>Page</u>
7.10	(a) Composite Oscilloscope Record Showing in Sequence from Top to Bottom, the Relative Delays between the Input Data Bit Pulse, and the Various Output Bit Pulses Corresponding to the Products Z_0 to Z_7 , respectively.	7-22
7.10	(b) Composite Oscilloscope Record showing in Sequence from Top to Bottom, the Input Data Pulse and Output Bit Pulses Corresponding to Z_0 (minimum delay) and Z_8 (maximum delay) for Maximum Operating Speed of the Multiplier (220 MHz).	7-22
8.1	General RNS Processor Configuration	8-5
8.2	Look-Up Tables for Multiplication and Addition in Modulo 5 Residue Arithmetic	8-7
8.3	Schematic Representation of a Modulo 7 Multiplication LUT with LED Light Sources as Look-up Indicators. The Numbers in each Block Represent the Detector Assigned to that Block.	8-9
8.4	Look-up Tables (Wiring Maps) for Additive and Multiplicative Inverses in Modulo 5 Arithmetic	8-10
8.5	Photograph of Prototype Modulo 7 LUT	8-13
8.6	Schematic Diagram of Prototype LUT	8-14
8.7	Mitsubishi ML 4402 Laser Diode Switching Characteristics	8-17
8.8	Prototype LUT Intensity Profile for the LUT Geometry (a), when the (b) (1,1), (c) (4,5), and (d) (7,7) Rows and Columns are Excited.	8-19
8.9	Responses of the 7 Laser Diodes of the First Row (a) and of the First Column (b)	8-20
8.10	Responses of the 7 Laser Diodes of the Sixth Row (a) and of the Sixth Column (b)	8-21
8.11	Responses of the 7 Laser Diodes of Column 4, When the (4,3) Laser Diode is Excited	8-23

LIST OF FIGURES (Cont'd.)

<u>Figure</u>		<u>Page</u>
8.12	B/R Conversion via Pipelined LUTs	8-25
8.13	Schematic Operation of 4-Residue Mixed Radix Decoder for Residue to Integer Transformation [after Tai et al. (Ref. 21)]	8-27
8.14	R/B Conversion Example	8-29
8.15	R/B Pipelined Conversion Example	8-30
8.16	Pipelined LUT Implementation of Eq. 8.3	8-31
8.17	Examples of Reduced Size LUT via the Use of Symmetry	8-34
8.18	Number of Laser Diode Rows (N_R) versus Number of Interconnections per Diode (n_I) for Modulo m_c	8-35
8.19	LUT Implementation of a Square Systolic Array for Matrix-Matrix Multiplication	8-40
9.1	Look-Up Table Set 1 Driven by \underline{w}_k to produce $1/\langle \underline{w}_k \underline{w}_k \rangle$ (see text for explanation of symbols)	9-12
9.2	Look-Up Table Set 2 Driven by μ_{i-i} , \underline{w}_k and $1/\langle \underline{w}_k \underline{w}_k \rangle$ to produce μ_k and $\mu_k \underline{w}_k$	9-13
9.3	Equivalent Block Diagrams for Look-Up Table Sets 1 through 4 and Delay Unit for Use in Subsequent Pipeline Processor Schematics	9-14
9.4	Pipelined Processor for Calculating the Elements of Q , E_i , and A^{-1} .	9-15
9.5	Pipelined Processor for Calculating the Products of E_i	9-17
9.6	Processor for Calculating $P = A^{-1}Q^*$	9-21
9.7	Array Processor for calculating $A^{-1} = EP$	9-22
A.1	Circularly Polarized Sampling Function	A-2
A.2	The Spectrum for a CP Sampled Signal	A-4

1. INTRODUCTION AND SUMMARY

The original objectives of this program were to explore optical methods for performing eigensystem calculations based on matrix-vector or matrix-matrix multiplications. Specifically, the program called for the analysis and design of a high accuracy Acousto-Optic (AO) vector-matrix multiplier together with pre- and post-processing electronics and input/output interfaces to implement an eigensystem solution algorithm suitable for optical implementation. The main goal of the program was to explore high accuracy (≥ 16 bits) optical-based special purpose systems whose performance would exceed, by orders of magnitude, the current or even projected performance of electronic systems such as CMOS, VHSIC, GaAs, etc. Such systems would find applications in the Adaptive Phased Array Radar (APAR) area, which by nature, has extremely high computational requirements, of the order of 10^{10} - 10^{12} M-A/s (multiplications/additions per second).

In the first phase of the program, available eigensystem-solution algorithms were studied, in order to determine their suitability for AO implementation (Section 2). The results of this study showed that all algorithms, aside from the matrix multiplication part, require a plethora of operations to be carried out electronically rather than optically. This is because optics cannot easily or practically perform either logical operations or certain arithmetic operations such as square roots and divisions. Such requirements make the possibly efficient use of the AO processor highly questionable. In addition, nearly all eigensystem or direct APAR algorithms require computational accuracies that exceed 16 binary bits. To accommodate such accuracies requires that the AO processor be incorporated in a system involving non-analog processing techniques. Unfortunately, there are not many algorithms suitable for implementing high accuracy multiplications/

additions with AO processors (Section 3). The only viable choice is the DMAC algorithm (Digital Multiplication via Analog Convolution) and its variations. Based on this algorithm, two novel AO architectures were developed, a single-detector, space-integrating AO system and a time-integrating, systolic AO system (Section 4). Utilizing state-of-the-art technology it was estimated that both systems could in principle deliver throughput rates of the order of 10^9 M-A/s. The performance of these systems was compared with that of state-of-the-art purely-electronic counterparts using as figures of merit the system efficiency, defined as throughput rate per unit power, and the multiplication speed (Section 6). A simple analysis showed that both systems (as well as other DMAC systems that have appeared in the open literature) do not offer any advantage over electronic systems that could be assembled with existing digital multipliers. The reasons for the relatively poor performance of the optical systems are: (1) the serial nature of DMAC and (2) the fact that optics does only part of the multiplication while the non-optical part relies on power-consuming A/D converters.

In view of this situation, we developed a Bit Parallel Multiplication technique (BPAM) for performing DMAC, which eliminates the serial problem (Section 3). Subsequently, we developed a novel space- and time-integrating BPAM AO system which is capable of performing multiplications in a single clock cycle (Section 4). However, an analysis showed that the multiplication speed and efficiency are only of the order of those already achieved with GaAs multipliers. The reasons for this are: (1) the nature of BPAM which incorporates an increased number of A/Ds and (2) the dimensions of the focused laser beam on the AO cell. Thus, we concluded that neither DMAC nor BPAM AO processors offer any significant advantage over existing electronic processors. However, in conjunction with DMAC, we developed a circularly polarizing sampling technique that allows for complex matrix operations with much reduced time and hardware constraints (Section 5). We believe that this

technique can be applied to any systolic or array processor including fully electronic systems.

These initial results were discussed with program monitors in meetings held in early 1985. As a result of these meetings, it was mutually decided to abandon all DMAC or BPAM AO related work and follow two new directions for the program that were related to the APAR problem. These two directions were: (1) optically interconnected electronic multipliers and (2) position-coded residue optoelectronic look-up table (LUT) processing.

The first topic involves optical interconnection techniques to enable high-speed multi-pin, electronic multipliers to be arranged in patterns that traditional micro-strip interconnects cannot handle (Section 7). For this purpose simple but efficient fiber-optic splitter/combiner techniques were studied and prototypes were developed in conjunction with a 4x4 bit fiber-optically addressed digital multiplier (Section 7.2). These results, although initial and largely incomplete due to shortage of both funds and time, show that existing low-cost fiber-optic technology can be used for globally interconnecting electronic array processors. We suggest that these ideas merit further development via the design and implementation of a fully electronic addressed square array processor.

The second direction involves the use of residue-based, high-speed LUTs that can be used for the APAR problem. Such LUTs allow for high-speed (single clock cycle) flexible operations such as multiplications, additions, subtractions and some forms of division. We approached this idea from both the LUT and APAR-LUT processor level. At the LUT level we suggested a novel implementation of a LUT which is based on the use of interlaced electrode laser diodes or light-emitting diodes (Section 8). This approach offers the advantage that existing technology can be used for the implementation of GHz-type operation

LUTs. We demonstrated the capabilities of the laser diode LUT by fabricating and testing a prototype. We show that data rates in excess of 250 MHz (RZ data) can be achieved even for discrete component LUTs. We believe that hybrid or monolithic approaches should offer switching speeds to allow data rates in excess of 1 GHz. Operation in the residue number system requires binary-to-residue and residue-to-binary conversions. With this in mind, we showed that LUT technology can be used in an efficient way for both conversions. Through an example of a typical residue LUT system, a square array processor, we showed that the conversions occupy about 20% of the hardware. This demonstrates the need to remain in the residue domain for as long as possible so that the conversion-required hardware is a relatively small fraction of the total. The LUTs require a large number of laser diodes even for moderate size applications and it is necessary to consider ways for hardware minimization both at the LUT and system levels. We examined LUT implementation techniques in which the number of laser diodes is reduced by about 50%. However, the corresponding number of interconnections required per laser diode is increased by a factor of two and we concluded that this is not a favorable approach. At the system level we studied a residue scaling technique which allows for scaling by factors of about 0.005% of the total dynamic range while maintaining accuracies of 9-10 bits. This seems to be a viable technique and further analysis is suggested in conjunction with specific applications.

For the APAR LUT problem we examined a variety of non-iterative algorithms for possible LUT implementation (Section 9). We found that the only choice that allows for residue LUT implementation is a variant of the Gram-Schmidt orthogonalization approach. In particular, this approach does not require square roots and it allows for the postponement of division until the last processing step. We showed, through examples, that this technique yields results identical to those obtained with straightforward arithmetic. We point out, however, that

this technique requires a rather large dynamic range which translates to excessive hardware. Based on this technique we designed a LUT-based pipelined processor which can invert a 6×6 APAR data matrix in about 7 clock cycles. This processor is a typical example of the flexibility afforded by the residue LUT approach. However, we emphasize that these are initial results and that much work is needed for further understanding of the LUTs, the algorithms, and the concept as a whole. We suggest that analyses are carried out to clearly demonstrate the competitiveness of the residue LUT approach compared with digital pipelined techniques.

2. EIGENSYSTEM SOLUTION

2.1 Introduction

In this section we consider the application of existing algorithms for determining the eigenvectors ϕ_n and eigenvalues λ_n of a matrix C where

$$C = \sum_{n=1}^N \lambda_n \phi_n \phi_n^T \quad (2.1)$$

One application for the eigenanalysis of a matrix is in a method for solving the Adaptive Phased Array Radar (APAR) problem illustrated in Figure 2.1. In this problem, which this program specifically addresses, we wish to calculate the adaptive weight vector \underline{w} which satisfies the system of equations

$$C\underline{w} = \underline{s} \quad (2.2)$$

where C is the data covariance matrix formed from M successive "snapshots" of the data vector $\underline{x}(m)$, i.e.,

$$C = M^{-1} \sum_{m=1}^M \underline{x}^T(m) \underline{x}(m) \quad (2.3)$$

and \underline{s} is the steering vector formed from the data vector $\underline{x}(m)$ and the reference signal x_0 , i.e.,

$$\underline{s} = M^{-1} \sum_{m=1}^M \underline{x}^T(m) x_0(m) \quad (2.4)$$

The solution to Equation (2.2) is written formally as

$$\underline{w} = C^{-1} \underline{s} \quad (2.5)$$

so that if the complete eigenvalues and eigenvectors of C are known, then from Equation (2.1)

$$\underline{w} = \sum_{n=1}^N \lambda_n^{-1} \underline{\phi}_n^T \underline{s} \underline{\phi}_n \quad (2.6)$$

The flow chart in Figure (2.1) shows the successive steps to be taken in determining the weight vector \underline{w} which is subsequently used to derive the antenna output signal $y(m)$ given by

$$y(m) = \underline{w}^T(m) \underline{x}(m) \quad (2.7)$$

To form the covariance matrix from the data vectors we require M outer products of size $N \times N$ (where N is the number of elements in the antenna) and M matrix additions. Once the covariance matrix is formed, a complete eigenanalysis of the matrix is made to determine the eigenvectors $\underline{\phi}_n$ and eigenvalues λ_n . The details of this eigenanalysis depend on the particular algorithm used; the different algorithms that can be used and which are particularly suited to optical implementation form the subject matter of the remainder of this section. Once the eigenvectors and eigenvalues are determined, the operations required for

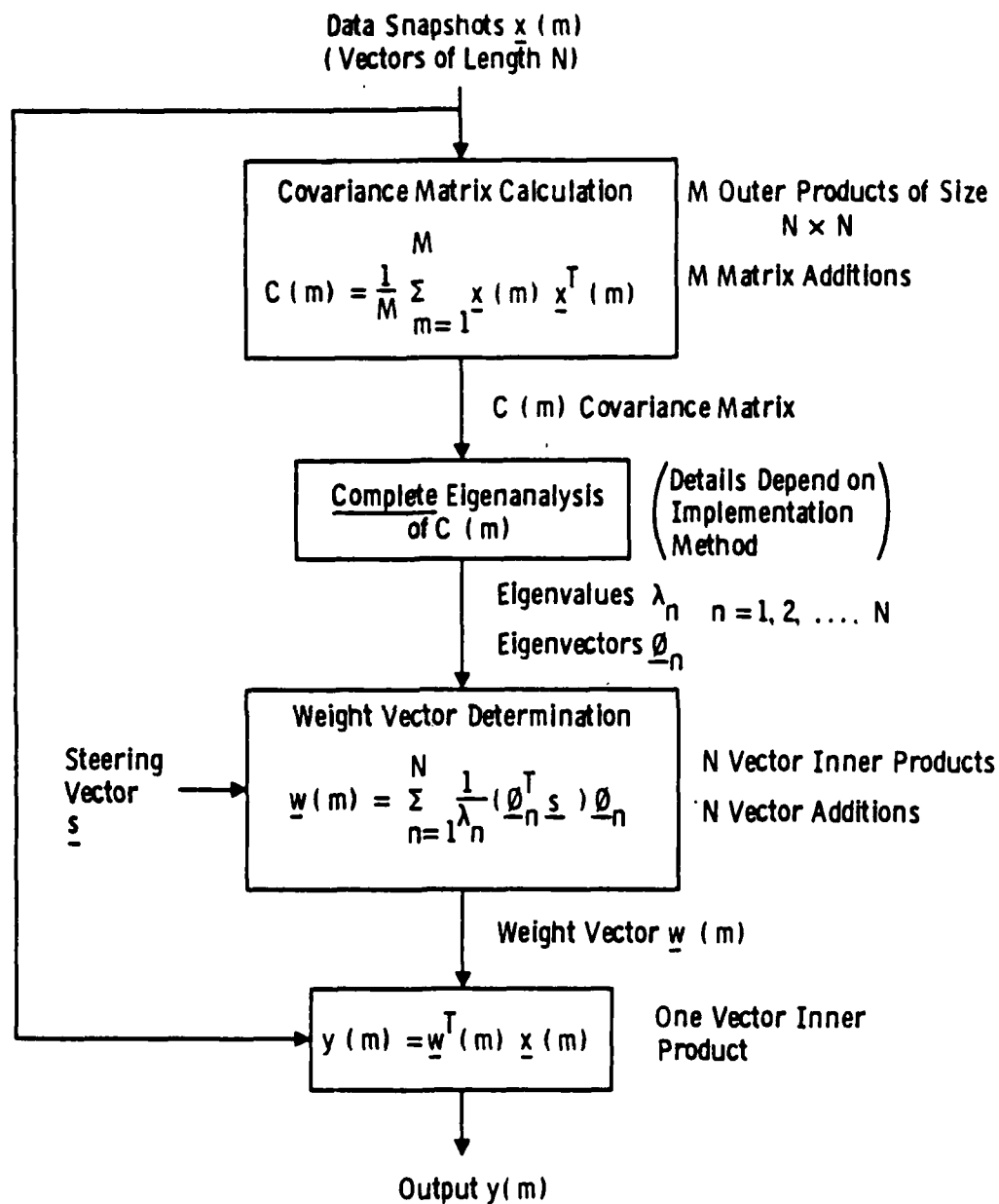


Figure 2.1 Weight Determination for APAR by Complete Eigenanalysis of the Data Covariance Matrix

forming the weight vector are N vector inner products and N vector additions. One further vector inner product is required to determine the output signal $y(m)$.

We note that all operations, leaving aside the eigenanalysis, are of the type which can readily be implemented optically. Thus, the focus for this method of solution of the APAR problem is on the algorithms available for eigenanalysis and, in particular, on those algorithms that are most suited to optical implementation.

2.2 Gershgorin Method

This method^{1,2} involves bounding the region of space in which the eigenvalues are located. The flow diagram used for this method is given in Figure 2.2. Thus, we first determine the radii, δ_i , of the discs which contain the eigenvalues, i.e.,

$$\delta_i = \sum_{\substack{j=1 \\ j \neq i}}^N |C_{ij}|, \quad 1 \leq i \leq N \quad (2.8)$$

The eigenvalues of C lie in the union of the discs D_i of radii, δ_i , centered at C_{ii} . If m of these discs are connected and disjoint from the remaining, these discs then contain exactly m eigenvalues of C . This requires a logic operation.

These bounds can be improved through repeated similarity transformations. Thus, the first Gershgorin disc D_1 is reduced by using $D C D^{-1}$ instead of C , where

$$D = \text{Diag} (\rho, \dots, 1) \quad (2.9)$$

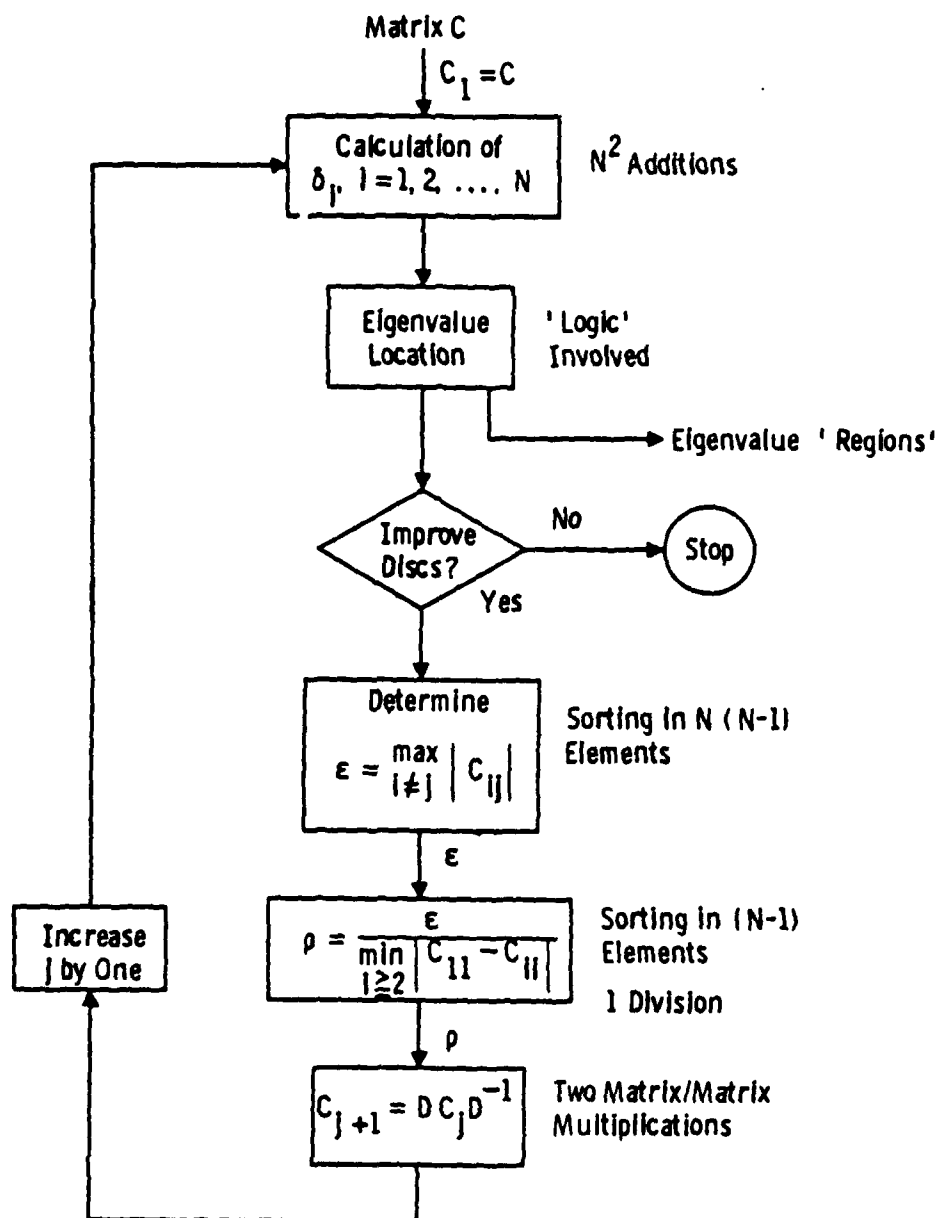


Figure 2.2 Flow Chart for the Determination of Eigenvalue Regions using the Gershgorin Method

$$\rho = \frac{\epsilon}{\min_{i \geq 2} |C_{11} - C_{ii}|} \quad (2.10)$$

and ϵ is the magnitude of the largest off diagonal element. To determine ϵ requires a sorting operation in $N(N-1)$ elements and ρ requires a similar sorting operation in $N(N-1)$ elements together with a division. Finally, the similarity transformation requires two matrix-matrix multiplications to derive $D C D^{-1}$.

The advantages of this method are that it is simple and it uses matrix-matrix multiplications, operations which can be implemented optically. However, the method finds only eigenvalue regions and logic is required for isolating the eigenvalues, an operation which optics cannot presently address. Further, there is no definite order in which the eigenvalues can be found. We also note that the method cannot be used to find eigenvectors. One procedure for finding eigenvectors from eigenvalues is that of inverse iteration, discussed in Section 2.4. Thus, the Gershgorin method may be useful as a first step for other methods such as inverse iteration.

2.3 Power Method

This method^{1,2} is an iterative method and is illustrated in Figure 2.3. We begin with some guess $\underline{z}^{(0)}$, for the dominant eigenvector and form

$$\underline{y}^{(1+1)} = C \underline{z}^{(1)} \quad (2.11)$$

in a matrix-vector multiplication operation. Next, the largest element of $\underline{y}^{(1+1)}$ is determined, which involves a sorting operation in N elements followed by N scalar multiplications, to derive

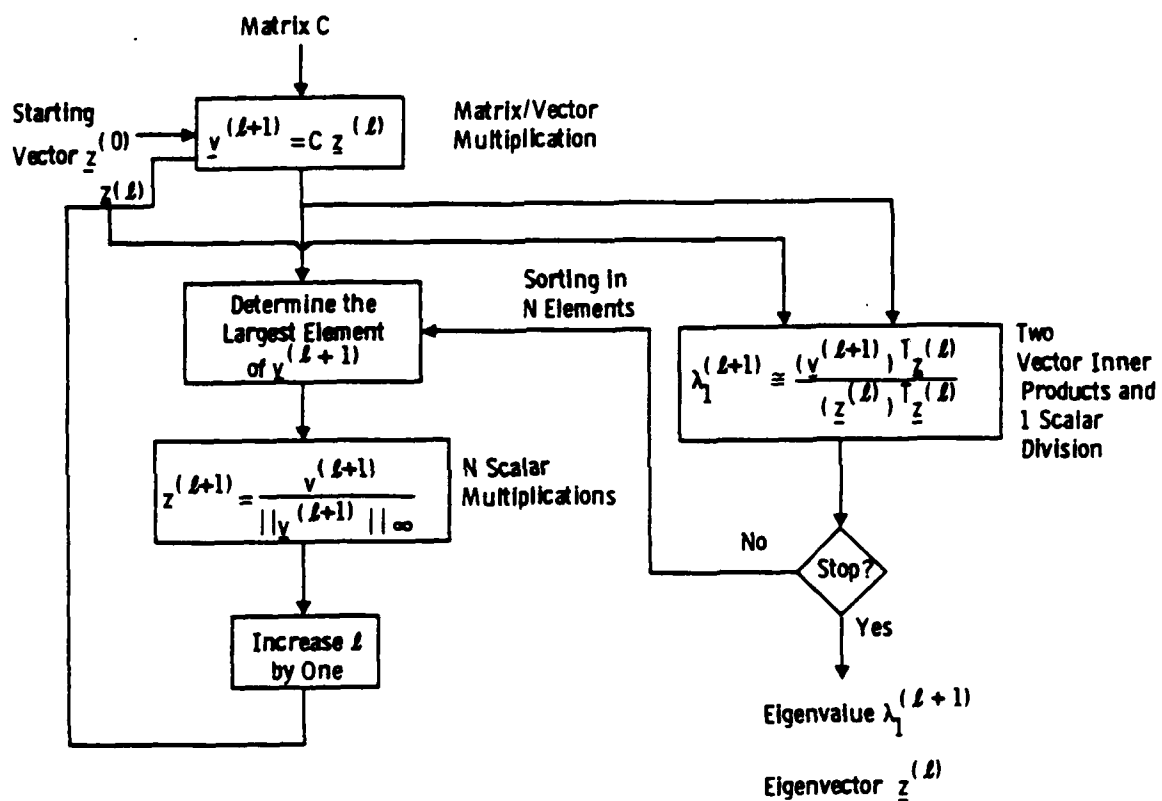


Figure 2.3 Flow Chart for the Determination of Eigenvalues and Eigenvectors using the Power Method

$$\underline{z}^{(1+1)} = \frac{\underline{v}^{(1+1)}}{\|\underline{v}^{(1+1)}\|_\infty} \quad (2.12)$$

As $l \rightarrow \infty$, we obtain the eigenvector estimate $\underline{\phi}_1$, i.e.,

$$\underline{z}^{(1)} \rightarrow \pm \frac{\underline{\phi}_1}{\|\underline{\phi}_1\|_\infty}$$

The corresponding eigenvalue estimate is given by

$$\lambda_1^{(1+1)} = \frac{[\underline{v}^{(1+1)}]^T \underline{z}^{(1)}}{[\underline{z}^{(1)}]^T \underline{z}^{(1)}} \quad (2.13)$$

The disadvantages of this method are that it is an iterative method requiring logic and division operations both of which cannot be readily implemented using optics. The convergence rate for obtaining solutions is

$$\lambda_1^{(1+1)} = \lambda_1 \left\{ 1 + O \left[\left| \frac{\lambda_2}{\lambda_1} \right| \right] \right\} \quad (2.14)$$

Various schemes for accelerating convergence have been proposed such as using a shift of origin, i.e., using $(C-pI)$ instead of C , using C^2 instead of C , and using the Rayleigh quotient $(\underline{z}_k^T C \underline{z}_k) / (\underline{z}_k^T \underline{z}_k)$.

The power method breaks down for complex eigenvalues with the same modulus.

2.4 Inverse Iteration

This method^{1,2} is a variation of the power method and is best known for finding the eigenvector corresponding to an eigenvalue close to some p . A flow chart for this method is given in Figure 2.4. Thus, knowing an approximate eigenvalue p , we first form, through N scalar additions, the matrix E given by:

$$E = C - pI \quad (2.15)$$

We then seek solutions to the equation

$$(C - pI) \underline{v}^{(l+1)} = \underline{z}^{(l)} \quad (2.16)$$

This is similar to the power method except that C is replaced by $(C-pI)^{-1}$, both having the same eigenvectors.

The next step involves the triangular decomposition of E to give

$$E = L U \quad (2.17)$$

The detailed operations involved in this decomposition are shown in the flow diagram of Figure 2.5, and involve scalar multiplications and additions together with logic decisions.

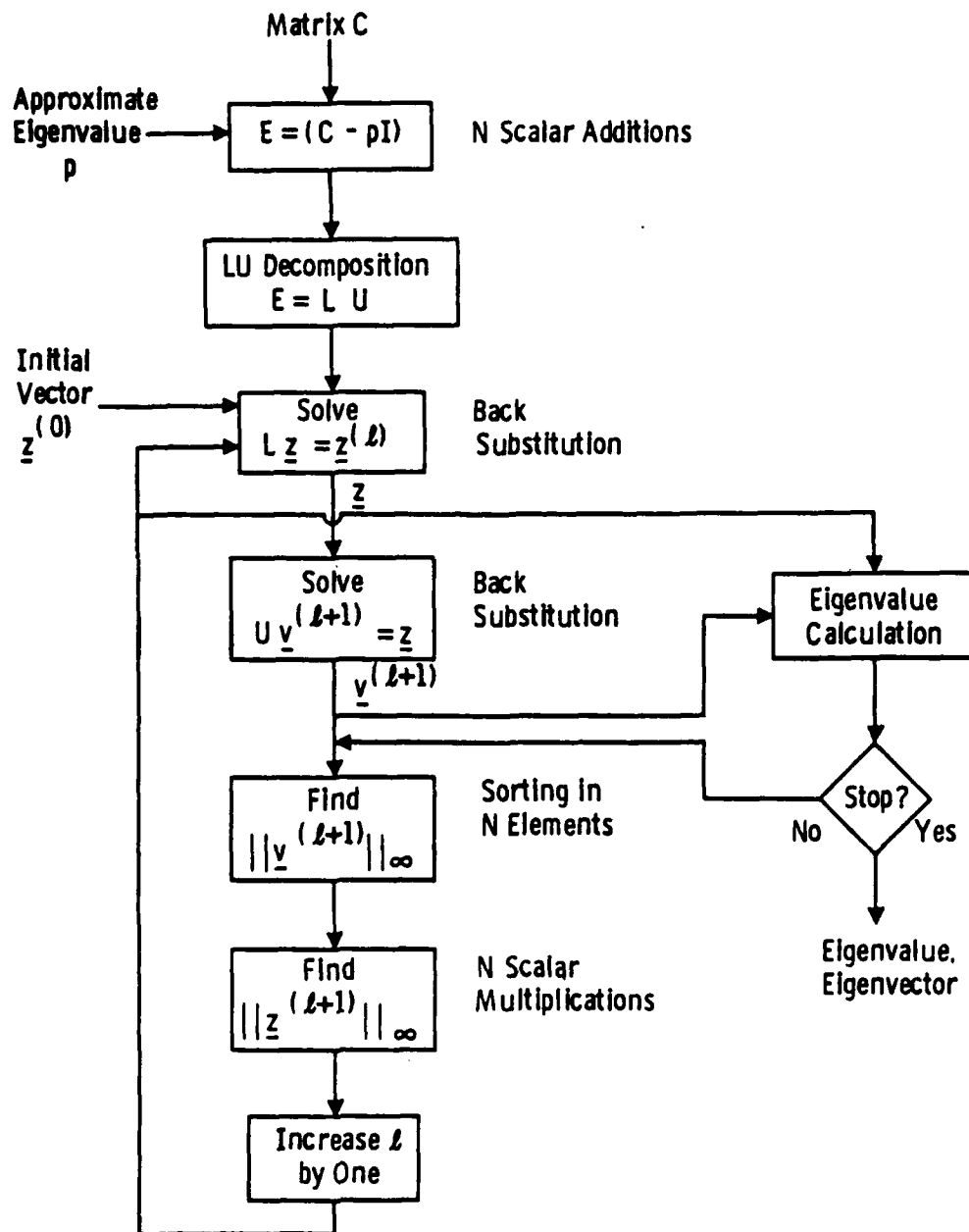


Figure 2.4 Flow Chart of the Determination of Eigenvectors Corresponding to Given Eigenvalues by the Method of Inverse Iteration

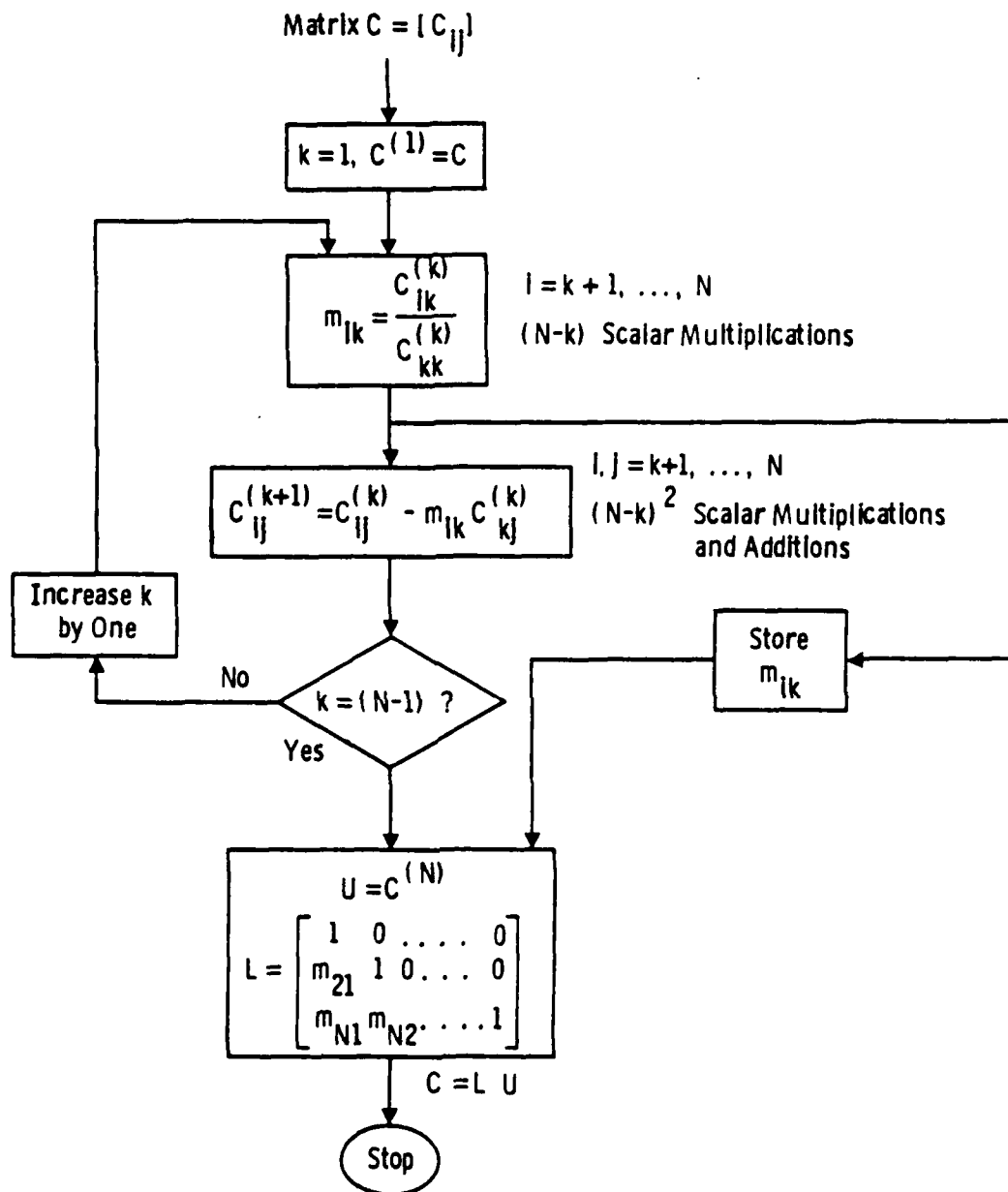


Figure 2.5 Flow Chart for the LU Triangular Decomposition of a Matrix

Having decomposed the matrix E we solve the sets of equations

$$L \underline{z} = \underline{z}^{(1)} \quad (2.18)$$

$$U \underline{y}^{(1+1)} = \underline{z} \quad (2.19)$$

by back substitution to derive the eigenvectors and eigenvalues. Operations involved here are sorting in N elements and N scalar multiplications.

Disadvantages of the technique are that a considerable number of transformations are required and the convergence properties are far from satisfactory. In addition, matrices exist which have no triangular decomposition in spite of the fact that their eigenproblem is well-conditioned, or whose triangular decomposition is numerically unstable.

2.5 Q-R Method

This method^{1,2} has proven to be the most effective of known methods of solving the general eigenvalue problem. In contrast to the method discussed in Section 2.4, it is based on unitary transformations. A flow diagram of the approach is given in Figure 2.8. The first step involves the Q-R decomposition of the matrix to give a factorization into the product of a unitary matrix Q and an upper triangular matrix R

$$C^{(1)} = Q^{(1)} R^{(1)} \quad (2.20)$$

The steps involved in this procedure are illustrated in Figure 2.7. Operations involve $(N-1)$ squares, $(N-1)$ additions, 1 division and 1 multiplication for each pass through the loop together with 1 outer product and matrix-matrix multiplication.

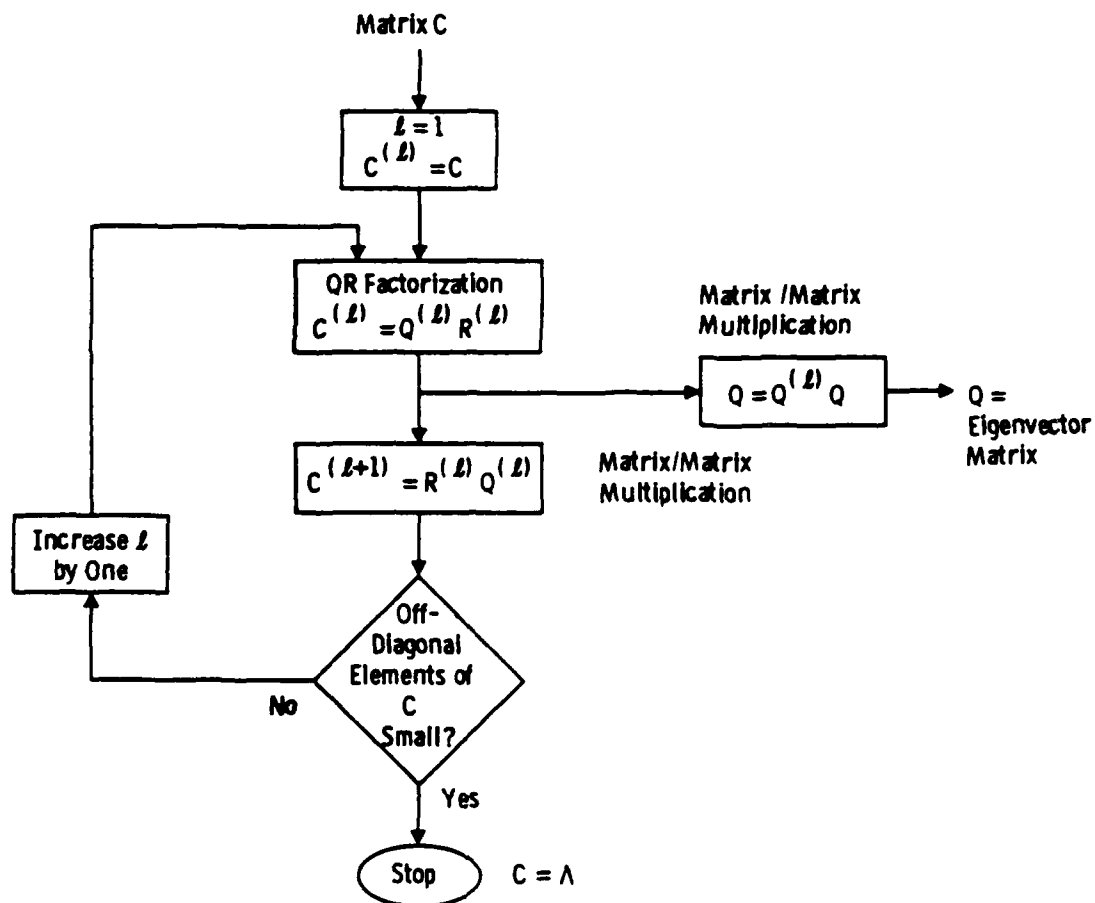


Figure 2.6 Flow Chart for Implementing the Q-R Algorithm

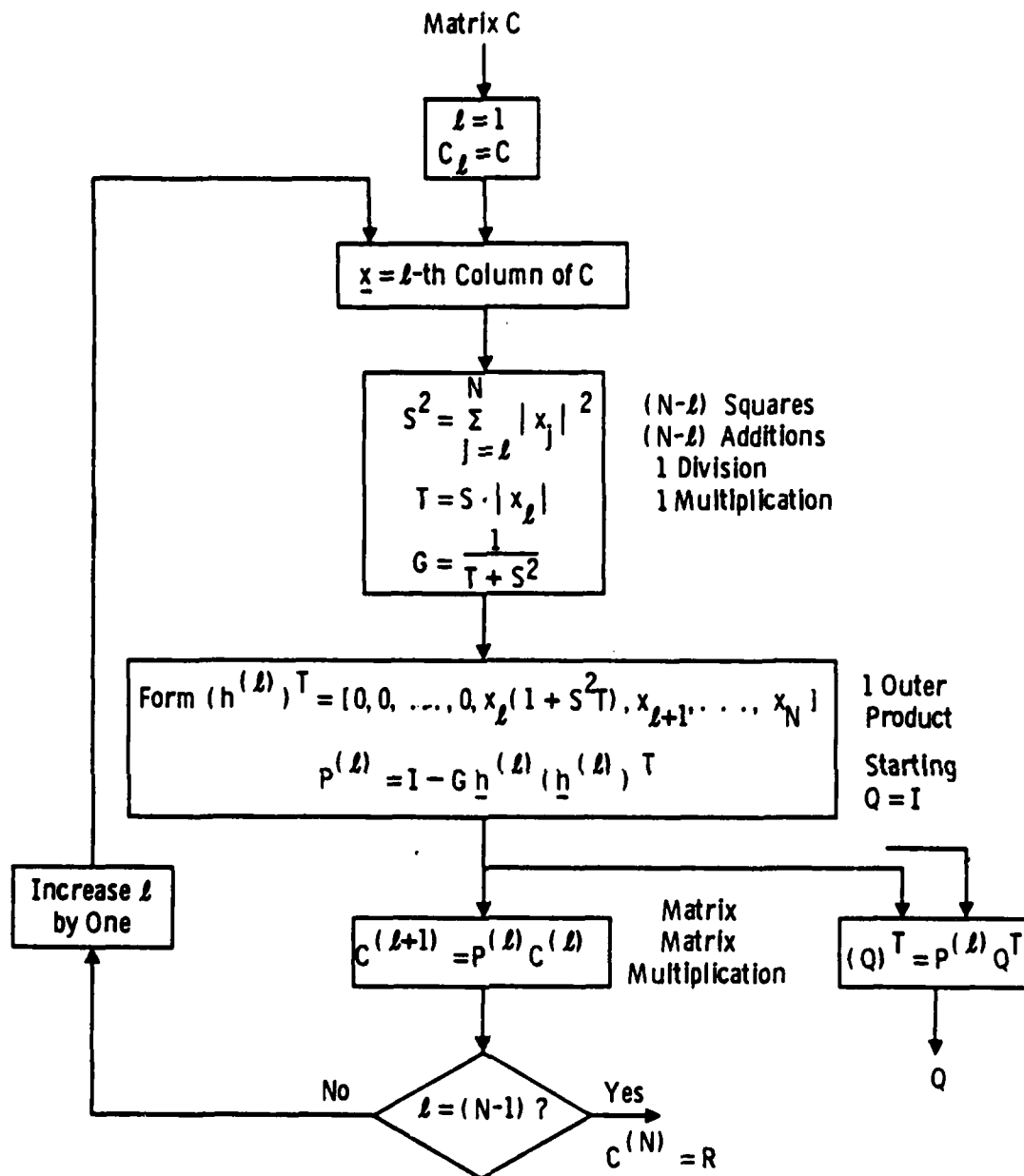


Figure 2.7 Flow Chart for the QR Factorization of a Matrix

From Equation (2.20) we form

$$C^{(l+1)} = R^{(1)} Q^{(1)} \quad (2.21)$$

$$= \begin{bmatrix} Q^{(1)} & Q^{(1-1)} & \dots & Q^{(1)} \end{bmatrix}^T C \begin{bmatrix} Q^{(1)} & Q^{(2)} & \dots & Q^{(1)} \end{bmatrix} \quad (2.22)$$

As $l \rightarrow \infty$,

$$C^{(l)} \rightarrow \text{Diag} (\lambda_1, \lambda_2, \dots, \lambda_N) \quad (2.23)$$

i.e., for large λ we have determined the eigenvalue matrix Λ

$$C^{(l)} = \Lambda$$

Thus,

$$\Lambda = Q^T C Q \quad (2.24)$$

or rearranging

$$C Q = Q \Lambda \quad (2.25)$$

where Q contains the eigenvectors.

The operations involve extensive matrix-matrix multiplication together with logic decisions.

2.6 Discussion

Although many algorithms exist for eigenanalysis of matrices, they all involve extensive arithmetic and logical operations. The basic arithmetic operations of addition, subtraction, and multiplication can be implemented using optics. However, operations such as division, square roots, and logic decisions cannot presently be implemented optically. Thus, the algorithms discussed in this chapter require, at best, a hybrid optical-electronic approach for use in a practical system.

For such a hybrid system, even if those operations which can be implemented optically prove to be executable at higher speed than can be obtained electronically, the overall speed is still dictated by non-optical operations. If continued switching between the optical and electronic domains is necessary as is the case for executing these algorithms, the value of the optical processing contributions to the overall system becomes questionable.

3. ALGORITHMS FOR HIGH-ACCURACY ACOUSTO-OPTIC PROCESSORS

Optical systems performing multiplication and addition can be implemented via analog, binary, and residue techniques. Analog processors, although fast, suffer from low accuracy and can only be used for applications where 8-10 bits accuracy is required. Due to the nature of our applications (eigensystem solution, APAR, etc.) we require accuracies well in excess of 10 bits (e.g., 16 or 20 bits) and, thus, we will not consider analog processors. This Section covers binary techniques which result in AO systems of high digital accuracy. Residue arithmetic methods will be discussed in Section 8.

3.1 Digital Multiplication via Analog Convolution (DMAC)

Multiplication of two binary numbers via analog convolution³ (DMAC) is based on the novel idea of convolving the binary words representing the two numbers. The result is generated in a mixed binary format where, like binary arithmetic, each digit is weighted by a power of 2; but unlike binary arithmetic, each digit can be > 1 . The algorithm can be best realized via some simple examples: consider the calculation of the products $15 \cdot 41$ and $29 \cdot 62$. We first convolve the binary representations of the numbers for each product. The results of the convolutions are

$$(15 \cdot 41): [0 \ 0 \ 1 \ 1 \ 1 \ 1] * [1 \ 0 \ 1 \ 0 \ 0 \ 1] = [0 \ 0 \ 1 \ 1 \ 2 \ 2 \ 1 \ 2 \ 1 \ 1 \ 1] \quad (3.1)$$

$$(29 \cdot 62): [0 \ 1 \ 1 \ 1 \ 0 \ 1] * [1 \ 1 \ 1 \ 1 \ 1 \ 0] = [0 \ 1 \ 2 \ 3 \ 3 \ 4 \ 3 \ 2 \ 1 \ 1 \ 0] \quad (3.2)$$

Next, we weight each convolution point by a power of 2. Finally we sum the weighted points to obtain the final result:

$$15 \cdot 41 = [1 \cdot 2^8 + 1 \cdot 2^7 + 2 \cdot 2^6 + \dots + 1 \cdot 2^0] = 615 \quad (3.3)$$

$$29 \cdot 62 = [1 \cdot 2^9 + 2 \cdot 2^8 + 3 \cdot 2^7 + \dots + 0 \cdot 2^0] = 1798 \quad (3.4)$$

Note that in order to sum different products (which is the case for inner products), we sum the corresponding points and subsequently weight and sum. For example:

$$(15 \cdot 41) + (29 \cdot 62) = [(1+0) \cdot 2^9 + (2+1) \cdot 2^8 + (3+1) \cdot 2^7 + \dots + (1+0) \cdot 2^0] = 2413 \quad (3.5)$$

The importance of this scheme is evident in considerations of dynamic range requirements. For example, to multiply two numbers each with a dynamic range of $N = 2^{16} = 65,536$, we need an output dynamic range of $N = 2^{32} = 4.3 \times 10^9$. With binary encoding, the input and output dynamic range must be 2 (i.e., "0" and "1") and 16 (i.e., when all 16 bits are "1"), respectively. Consider now the summation of 50 such products. If analog techniques were used, we would need an output dynamic range of 2.1×10^{11} . With the binary scheme, we need input and output dynamic ranges of 2 and $50 \times 16 = 800$.

Notice that once the convolution data have been generated (in analog form), an A/D converter in conjunction with a shift-register/accumulator can be used to convert the mixed binary data to binary data. The A/D converter requirement is for $\log_2 k$ bits, where k is the maximum value of the mixed binary data.

The DMAC technique can be extended, via a twos complement encoding, to handle both positive and negative numbers. To allow sign notation, the leftmost bit for each binary word is the sign bit: 0 for plus and 1 for minus. Positive binary numbers are represented by their original binary form with the addition of the sign bit. For example, the integer +13 is represented by 0 1 1 0 1.

To represent a negative number we first change the sign bit of its signed binary absolute value from 0 to 1. Next, we change all the ones to zeros and all zeros to ones (this is the ones complement

representation). Finally, we add a 1. For example, for the integer -45, the ones complement representation is 1 0 1 0 0 1 0, which in two complement form becomes 1 0 1 0 0 1 1. Conversion from the twos complement representation to signed absolute value is obtained by changing all zeros (ones) to ones (zeros) and adding a 1.

As is shown in Reference 4, one technique of multiplying two numbers, using twos complement binary representation, requires that the input numbers be represented by the same number of bits required to represent the output. For example, consider the product $+13 \times -45 = -585$. To represent the output, we require a total of 11 bits, including the sign bit. To extend the input numbers to 11 bits, we insert six zeros between the sign bit and the most significant bit (MSB) of +13 and four ones between the sign bit and the MSB of -45. Thus the input numbers become 0 0 0 0 0 0 0 1 1 0 1 and 1 1 1 1 1 0 1 0 0 1 1 for +13 and -45 respectively. The product of the binary numbers can now be calculated by performing a usual-sense multiplication with the exception that any bits generated to the left of the sign bit column are truncated. An example of this procedure is shown in Figure 3.1 for the case of the product $+13 \times -45$. The result is expressed in a mixed binary form and it can be converted to twos complement representation by: divide the least significant mixed binary bit by modulus 2, add the quotient to the next bit, divide by 2, add the quotient to the next bit, divide by 2, add the quotient to the next bit.....etc.

The remainders of these series of operations constitute a binary word which is the twos complement representation of the mixed binary output. Note that the remainder of the first division is the LSB and the remainder of the last division is the sign bit of the so-obtained twos complement binary word. An example of this procedure is shown in Figure 3.2 for the case of the product $+13 \times -45 = -585$.

(+13)	0	0	0	0	0	0	0	1	1	0	1	Twos Complement Representation
(-45) X	1	1	1	1	1	0	1	0	0	1	1	
	0	0	0	0	0	0	0	1	1	0	1	
	0	0	0	0	0	0	1	1	0	1		
	0	0	0	0	0	0	0	0	0			
	0	0	0	0	0	0	0	0				
	0	0	0	1	1	0	1					
	0	0	0	0	0	0						
	0	1	1	0	1							
	1	1	0	1								
	1	0	1									
	0	1										
	1											
	3	3	2	2	2	0	2	2	1	1	1	Mixed Binary Representation

Figure 3.1 Example of Multiplication in Twos Complement Representation

(1)	mod 2 = 0	R = 1	(LSB)
(1 +	↓) mod 2 = 0	R = 1	
(1 +	↓) mod 2 = 0	R = 1	
(2 +	↓) mod 2 = 1	R = 0	
(2 +	↓) mod 2 = 1	R = 1	Twos Complement Representation
(0 +	↓) mod 2 = 0	R = 1	
(2 +	↓) mod 2 = 1	R = 0	
(2 +	↓) mod 2 = 1	R = 1	
(2 +	↓) mod 2 = 1	R = 1	
(3 +	↓) mod 2 = 2	R = 0	
(3 +	↓) mod 2 = 2	R = 1	(MSB) (Sign Bit)

Figure 3.2 Example of Conversion from Mixed Binary Representation to Twos Complement Representation

As Figure 3.2 suggests, the result is 1 0 1 1 0 1 1 0 1 1 1 which is the twos complement representation of the number -585. Note that if the mixed binary result is generated in the form of an analog signal, then the device necessary for the conversion is an A/D converter followed by a shift register/accumulator.

As mentioned earlier, the mixed binary form of the output allows addition of different products without the need for carries. To illustrate this, consider the summation of the products (+13 x -45) and (+13 x -10). The mixed binary representation of the product +13 x -10 = -130 is 3 3 3 3 2 2 3 1 1 1 0. Addition of this result to the one that corresponds to the product +13 x -45 yields

$$\begin{array}{r}
 3\ 3\ 2\ 2\ 2\ 0\ 2\ 2\ 1\ 1\ 1 \\
 +\ 3\ 3\ 3\ 3\ 2\ 2\ 3\ 1\ 1\ 1\ 0 \\
 \hline
 6\ 6\ 5\ 5\ 4\ 2\ 5\ 3\ 2\ 2\ 1
 \end{array}$$

Conversion of the mixed binary result to twos complement gives 1 0 1 0 0 1 1 0 1 0 1 which is the twos complement representation of the number (-585) + (-130) = -715.

From the above brief discussion it is apparent that incorporation of the DMAC-twos complement arithmetic by optical processors, solves two major problems; specifically, accuracy and bipolar number handling. However, note that the use of such algorithms results in a major sacrifice in the processor's time-bandwidth product (TBW), specifically, a reduction by at least a factor of $2N-1$, where N is the number of bits in the input. This is due to the nature of the serial-type convolution which requires $2N-1$ clock cycles for its completion. Note that aside from the TBW reduction, we undergo a multiplication speed reduction (as compared to the clock). Specifically, the digital multiplication time required is $T_b \times (2N-1)$

where T_b is the clock pulse-width. These issues are discussed in more detail in Section 6.

3.2 Bit Parallel Multiplication (BPAM)

To avoid the DMAC problems, we have developed a bit-parallel digital multiplication (BPAM) algorithm.⁵ This can be explained via a simple convolution example. Suppose we wish to convolve the sequences $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ where A_i, B_i $i = 0,1,2,3$ are digits of binary value. Since $N=4$, we should obtain $2N-1 = 7$ convolution points. A rigorous implementation of the convolution, shows that the 7 convolution points are

$$\begin{aligned}
 P^0 &= A_0B_0 \\
 P^1 &= A_1B_0 + A_0B_1 \\
 P^2 &= A_2B_0 + A_1B_1 + A_0B_2 \\
 P^3 &= A_3B_0 + A_2B_1 + A_1B_2 + A_0B_3 \\
 P^4 &= A_3B_1 + A_2B_2 + A_1B_3 \\
 P^5 &= A_3B_2 + A_2B_3 \\
 P^6 &= A_3B_3
 \end{aligned} \tag{3.6}$$

From Eq. (3.7) we can observe the following:

- (1) The output convolution points (P^0 through P^6) are linear combinations of various A_iB_j products (e.g., $P^1 = A_1B_0 + A_0B_1$).
- (2) If all A_iB_j products are available in parallel, one can form the output convolution points by summing properly.
- (3) If the products and the various product summations can occur in parallel, then the time required for digital multiplication is no longer $2N-1$ clock cycles but rather 1 clock cycle.

From the above we see that, in principle, a BPAM can be formed in a single clock cycle as long as all the $A_i B_j$ input bit combinations are available in parallel. Note that the summation of M different number products (i.e., inner product) can be achieved in a way similar to the one for DMAC; i.e., sum in parallel all the $A_i B_{jm}$, $i, j = 1, 2, \dots, N$, $m = 1, 2, \dots, 2N-1$ convolution points.

The BPAM approach solves at least two major problems (as compared with DMAC); namely, (1) TBW reduction and (2) net multiplication speed. However, it creates a problem which is absent in DMAC; namely, it requires $N \times N$ output points for a single multiplication, which translates to a high output resolution requirement. Nevertheless, it is the only binary technique that guarantees both high speed and accuracy.

4. OPTICAL ARCHITECTURES FOR DMAC AND BPAM

In this Section we discuss a number of possible optical architectures which we have developed, in conjunction with the algorithms of Section 3. The first family of processors (space-integrating Acousto-Optic processor and time-integrating Acousto-Optic processor) represent systems that are based on the serial-type convolution. The performance of these processors is typical of that expected from systems which utilize serial DMAC. These processes can be fabricated using present custom technology. The second family of processors (BPAM Acousto-Optic processor) represent systems that are based on bit-parallel digital multiplication (BPAM). Unlike the first class of systems (which require $2N-1$ clock cycles for the formation of the convolution) the multiplication is formed in a single clock cycle thereby greatly enhancing the net multiplication speed. These architectures are good examples of BPAM optical processors that can be fabricated with present technology. Furthermore, these architectures should serve as a guide to the performance that may be expected from a BPAM processor.

4.1 DMAC Acousto-Optic Space-Integrating Processor

A simple binary number multiplication can be achieved using Acousto-Optic (AO) techniques and the serial convolution scheme of Section 3.1. Suppose we want to multiply two numbers; A and B, each represented by N bits. The bits of both numbers are made available serially (as a function of time) and named $S_A(t)$ and $S_B(t)$, respectively.

The optical system shown in Figure 4.1 is composed of two AO cells arranged in a counter-propagation configuration (i.e., the sound waves travel in opposite directions). The first-order diffracted beam from AO1 is imaged on AO2 through a pair of lenses and a spatial filter

Dwg. 9354A36

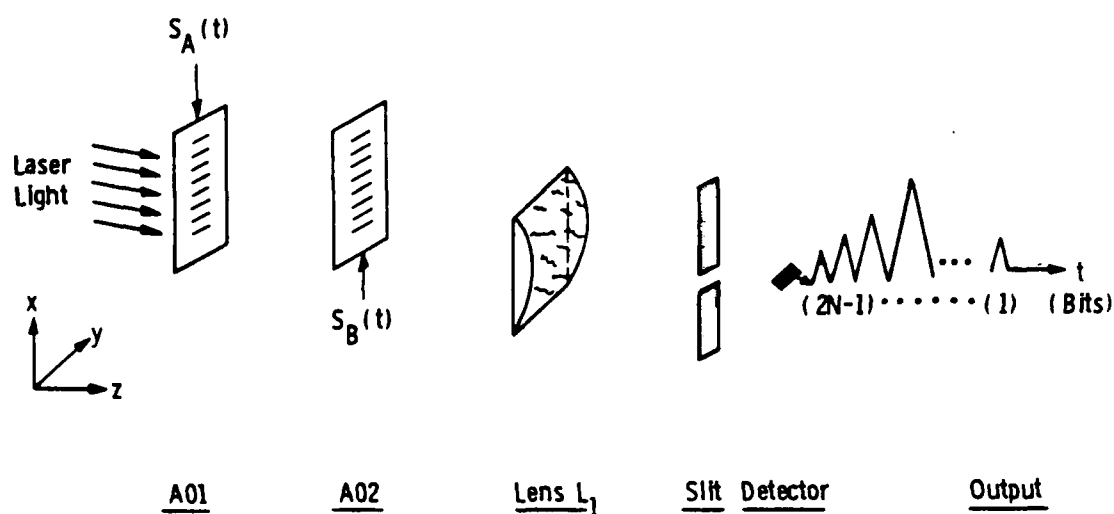


Figure 4.1 Acousto-Optic Processor for Binary Number Multiplication

(not shown). The product of the resulting distributions is imaged onto cylindrical lens L_1 through a second pair of lenses and a spatial filter. A detector follows a slit, which is placed at the back focal plane of L_1 , so that only the DC part of the resulting Fourier Transform is detected. Thus, the system part L_1 -slit-detector is used in order to form the integration of the product of the data present in A01 and A02. The data $S_A(t)$ and $S_B(t)$ are applied simultaneously onto the A0 cells. At every instant of time, lens L_1 forms the summation over the bit-by-bit products $S_A \cdot S_B$. The resulting light is detected by the detector. Because the data $S_A(t)$ and $S_B(t)$ are moving in opposite directions, the light incident on the detector is proportional to the convolution $S_A(t) * S_B(t)$. Consequently, the output of the detector is proportional to the convolution values. Since the data S_A and S_B are composed of N bits, the convolution is composed of $2N-1$ parts each triangular in shape, under the assumption that the bits are represented by square pulses. Note that the maximum value of the convolution occurs when all S_A and S_B data are present in the A0 cells. This corresponds to the highest triangle of Figure 4.1. Thus, we see that the simple arrangement of Figure 4.1 performs the first step of the binary algorithm; namely, the convolution. To obtain the product $A \cdot B$, we have to weight each convolution point by a power of 2 and then sum the results. This can be achieved if an A/D converter follows the detector and feeds its output to a digital shift-register/accumulator. When all $2N-1$ convolution parts have been accumulated, the values of the shift register are read out. This binary word corresponds to the product $A \cdot B$ with an accuracy of $2N$ bits (each number is represented by N bits in the input).

We can now expand the binary number multiplier system of Figure 4.1 to a multi-channel system for vector-matrix multiplication. Suppose we want to multiply the vector $b_{11}, b_{21}, \dots, b_{M1}$ with the matrix A consisting of elements a_{ij} , where $i = 1, 2, \dots, M$ and $j = 1, 2, \dots, M$. The result will be M inner products C_{i1} , where $i = 1, 2, \dots, M$.

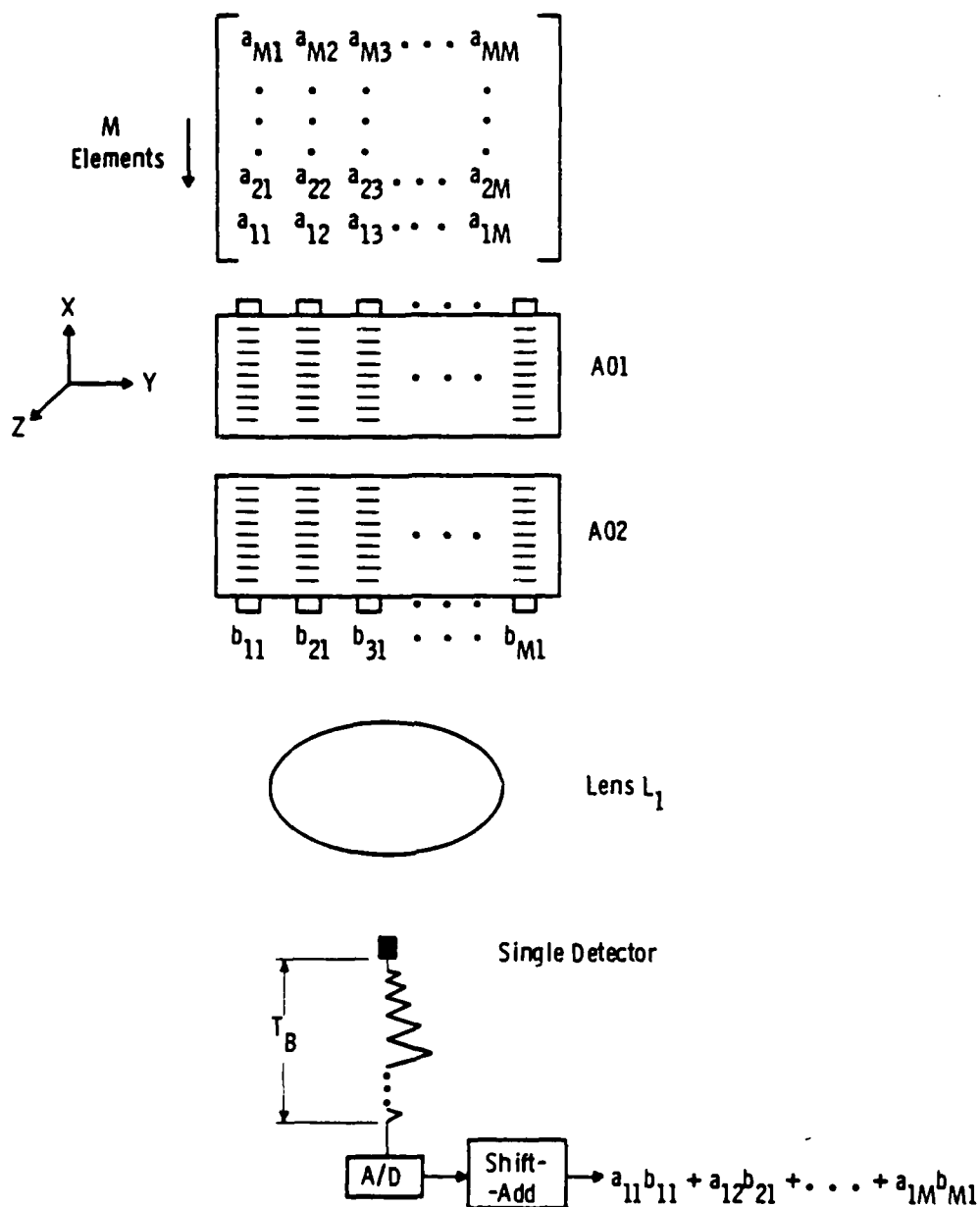


Figure 4.2 Space Integrating Acousto-Optic Processor for Accurate Vector-Matrix Multiplication

The optical system shown in Figure 4.2 is similar to the optical system of Figure 4.1, except that: (1) A01 and A02 are M-channel cells and (2) lens L_1 is spherical rather than cylindrical. If a cylindrical lens is substituted in place of L_1 then the system is an exact multi-channel version of the system of Figure 4.1. Thus, M detectors placed in parallel at the focal plane and at locations $x = 0$, $y = y_1$, $y = y_2$, ..., $y = y_M$ (where $y_1, y_2, y_3, \dots, y_M$ are the locations of the M AO cell channels across y), record M parallel convolutions:

$$a_{11} * b_{11}, a_{12} * b_{21}, \dots, a_{1M} * b_{M1} \quad (4.1)$$

If these convolutions are added and weighted, the result corresponds to a single inner product, C_{11} . This is exactly the function of the power of the spherical lens L_1 along y. Thus, the lens L_1 accomplishes two tasks: (1) it performs the necessary convolution integral along x and (2) it sums the various convolution points along y similar to the operation shown in the parentheses of Eq. (3.5). This operation is allowed because of the mixed-binary format of the output (resulting from the binary multiplication scheme) which allows for product summation without the need for carries. Thus, the output of the detector corresponds to a convolution which, after the required post-processing, is equal to the inner product:

$$C_{11} = [a_{11}b_{11} + a_{12}b_{21} + \dots + a_{1M}b_{M1}] \quad (4.2)$$

To obtain the second inner product C_{21} , the vector $b_{11}, b_{21}, \dots, b_{M1}$ is loaded onto A02 while the vector $a_{21}, a_{22}, \dots, a_{2M}$ is loaded onto A01. This procedure is repeated M times until, all $C_{ij}, i = 1, 2, \dots, M$ inner products (that correspond to the vector-matrix product $A \cdot B$) are obtained. Similarly, and in order to obtain a full matrix-matrix multiplication, this procedure is repeated $M \times M$ times.

We see that the system of Figure 4.2 is a real-time high-accuracy vector-vector multiplier which can be used for either vector-matrix or matrix-matrix multiplication. From the systems' point of view, the above multiplier offers another very significant advantage, namely, a single output, since a single detector is used for inner product detection. Consequently, the required interface with a digital microprocessor is very simple.

It is worthwhile to mention that if M detectors are used (in conjunction with a cylindrical lens, L_1) the system can still calculate inner products. In this case the necessary product summation must be carried out digitally. This obviously increases the complexity of the electronic post-processing as well as of the interface, but it offers some additional flexibility. Specifically, it allows for separate operations over the various input/output channels, just like a conventional digital array processor. Whether one wants to use a processor with a single detector or M detectors is a question that depends on the specific algorithms used and can be answered only when a specific analysis of existing algorithms is made in conjunction with the architectural choices.

4.2 DMAC Acousto-Optic Space-Integrating Processor Characteristics

The binary algorithm allows the processor to have input and output dynamic ranges of N and $2N$ bits, respectively. The component dynamic range requirements are 2:1 for the AO cells and $N \times M$:1 for the detector in a single-detector system or N :1 in a multi-detector system. This is because, for full inner product formation, the maximum possible value of the output convolution is $N \times M$, which occurs when all M input numbers have all their input bits at logic "1". Consider now a specific example of $M = 128$ and $N = 8$ (16 bits output). Then $N \times M = 1024$, which means that for a single detector system, the dynamic range of the detector needs to be at least 1024:1. In practice, however, the dynamic range

of the detector should be higher in order to avoid detection errors (which might be severe if one takes into account the post-processing stage where each convolution point is weighted by a factor of 2). A simple statistical analysis shows that,⁶ in order to keep the bit error probability to $< 2.9 \times 10^{-7}$, the dynamic range of the detector needs to be increased by a factor of 10, which corresponds to 10,000:1. This requires a detector with 40 dB dynamic range, which is commercially available. On the other hand, if a 128-detector system is to be used, the maximum value of the convolution is N, which corresponds to a detector dynamic range of 10 x 8:1 or 19 dB. It is evident that the detector requirements are not severe and can be met with commercially available devices.

To avoid computational errors, both A0 cells should be very uniform over their entire apertures. This requirement is significant, especially for the single detector system, and comes about because of the 2-D spatial integration used. If non-uniform devices are used, convolution points of the same analog values will correspond to different light levels, and spatial integration will consequently yield an incorrect output.

Initial analysis shows that the uniformity required should be better than N x M:1 and tends to approach the dynamic range of the detector(i.e., 10 x N x M:1). The uniformity requirement of the A0 cell is achievable because the required A0 cell time-aperture is relatively small (i.e., for N = 9 and bit-width of 10 nsec, the required aperture is 0.08 μ sec). The small apertures associated with low acoustic attenuation A0 crystals result in a very uniform acoustic field, which corresponds to the propagating bit-stream. On the other hand, because of the small aperture, acoustic diffraction is controllable. Note that the effects of acoustic diffraction, in conjunction with the algorithm used, can be severe (this is explained in detail in Reference 7).

The A/D converter requirement is $\log_2(N \times M)$ bits for a single detector system or $\log_2(N)$ bits for an M-detector system. For example: for $N = 8$ and $M = 128$, $\log_2(N \times M) = 10$ and $\log_2(N) = 3$. These A/D requirements are easily met with commercially available devices.

The throughput rate of the system depends on several factors: (1) number of input channels, (2) number of output channels, (3) number of input bits, (4) input bit width and (5) number and speed of available A/D converters. Let us calculate the throughput rate of the system based on rather optimistic data. We assume the availability of a Bragg cell with 128 input channels and use 128 output channels (i.e., the most flexible version of Figure 4.2 where lens L_1 is cylindrical).

For $N = 8$ and a bit width $T_B = 100$ nsec, the total time required for formation of a single inner product is

$$T = (2N-1) T_B = 1.5 \mu\text{sec} \quad (4.3)$$

where the extra $(N-1)T_B$ time represents the total duration of zeros which follow the N bits. This is required in order to separate the different inner products. During this time, the system (with $M = 128$) has performed 128 multiplications. Thus, the throughput rate of the system is

$$R = 128 M-A / 1.5 \times 10^{-6} \text{ sec} = 8.5 \times 10^7 \text{ M-A/sec} \quad (4.4)$$

To improve the throughput rate of the system, we need to decrease T , which implies that we need to decrease the input bit width. For example, for $T_B = 3$ nsec, the throughput rate of the system is $R = 2.84 \times 10^9$ M-A/sec or 2.8 GOPS.

For this scenario with $T_B \approx 3$ nsec, the output A/D requirements are:

- (1) 128 3-bit A/D's with a speed of 300 MHz (for the multichannel output

version) and (2) a single 10-bit A/D with a speed of 300 MHz (for the single output version). Clearly the single-channel output version is impractical since 10-bit A/D's at 300 MHz are not presently available. On the other hand, the multi-channel output version, although difficult, is more realistic.

4.3 DMAC Acousto-Optic Time-Integrating Processor

The basic unit of this processor is the classical time-integrating AO processor whose schematic diagram is shown in Figure 4.3. We first describe the operation of the unit for the formation of a single product (e.g., $+13 \times -45$) via the twos complement scheme.⁴ The AO cell is driven by the binary data $a = +13$ in a bit-serial mode (Figure 4.4). The sign bit is applied first. At time $t = t_1$ all bits that correspond to the number $+13$ have been loaded into the cell. The so-created spatial distribution is Schlieren imaged onto a time-integrating linear detector array. The array consists of N elements, where N is the number of output bits (e.g., for our example $N = 11$). At time $t = t_1$ the binary data that correspond to the number $b = -45$ are applied onto the laser diode in a bit-serial mode. These data are applied such that the LSB is first. The resulting pattern is time-integrated by the detector array. At time $t = t_2$ the data in the AO cell have moved by distance d_B which corresponds to a time-delay T_B equal to the duration of a bit plus a zero (i.e., $T_B = t_2 - t_1$, see Figure 4.4). At the same time the second bit (i.e., $\text{LSB} + 1$) of number b is applied onto the laser diode. A new pattern is created which is added, by the detector array, onto the already existing pattern. At time $t = t_3$, a new pattern leaves the AO cell, is added by the detector, etc. Thus, after time $t = t_{11} - t_1 + T_B$ the last (i.e., 11th) pattern has been created and added. At this point, the values of the N elements of the detector are read. A close inspection of Figure 4.4 shows that the readout charge has a value (function of element) that corresponds to 3 3 2 2 2 0 2 2 1 1 1, which is the mixed binary form of the number $+13 \times -45 = -585$. These analog

Dwg. 9358A98

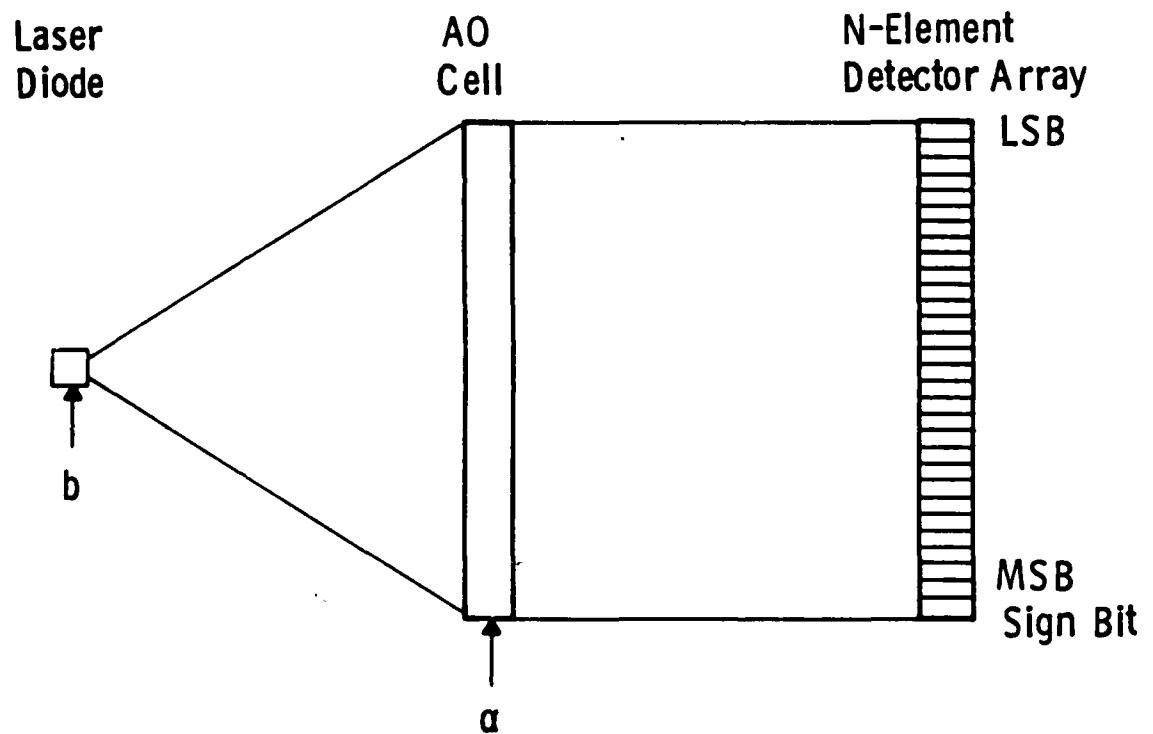


Figure 4.3 Time-Integrating Acousto-Optic Processor

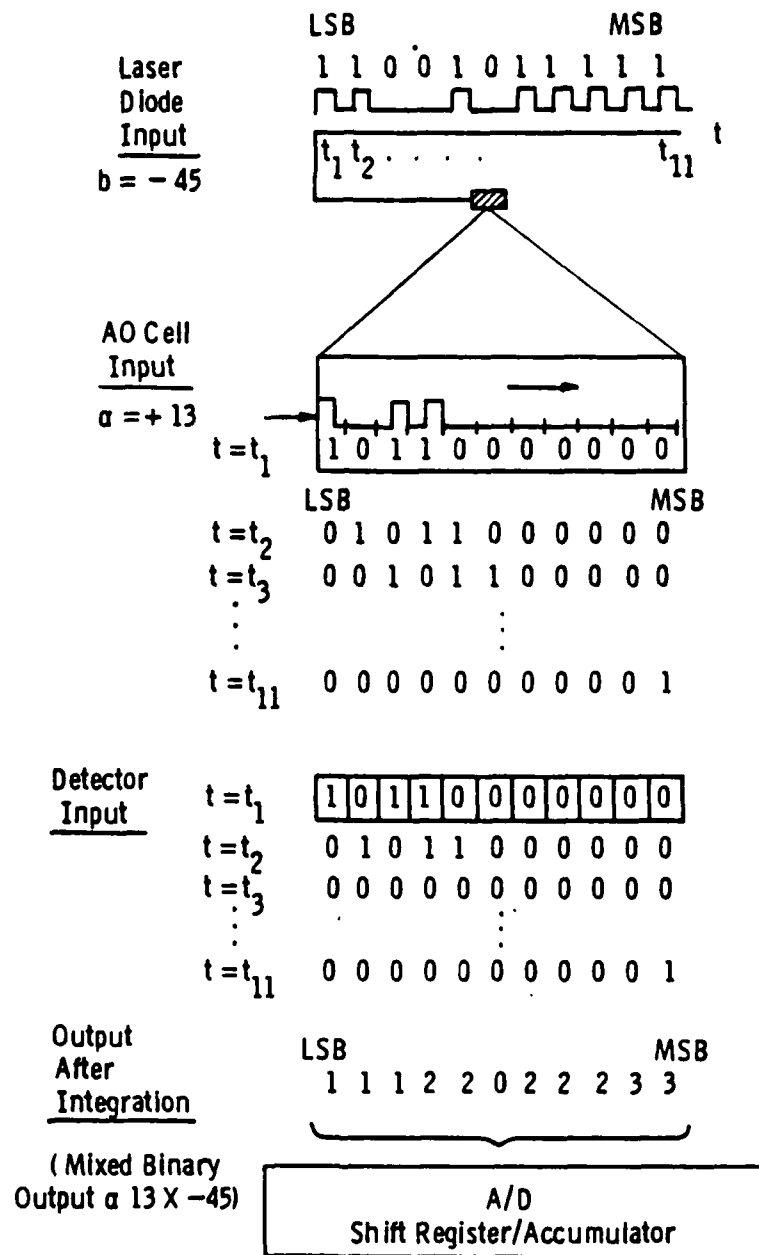


Figure 4.4 Input Timing Diagram for the Processor of Figure 4.3

values are consequently converted into twos complement representation via the A/D/shift-register/accumulator unit.

We emphasize two important points: (1) each bit in either a or b data is followed by a zero. This is necessary in order to separate (onto the detector array) different output bits, and is due to the fact that while the data b are applied, data a are moving. (2) the LSB of the word a is followed by $2N$ zeros of total duration $T = NT_B$. This is necessary in order to distinguish between different products; that is, if the LSB of a is followed by the sign bit of a number c , the output would be incorrect due to contributions from the product bc .

The unit we have described serves as a simple two-number multiplier. Extension to vector-vector multiplication (via inner product formation) can be achieved via the multi-unit architecture of Figure 4.5. In this case an array of M laser diodes in conjunction with an M -channel AO cell is used. Each of the a_i ($i = 1, 2, \dots, M$) elements of vector \underline{a} drives a different AO cell channel. Similarly, each of the b_i ($i = 1, 2, \dots, M$) elements of vector \underline{b} drives a different diode.

For the time being let us ignore the cylindrical lens. Instead, let us assume that the AO cell is imaged onto an $M \times N$ element, 2-D detector array. This system is basically a multi-channel version of the system of Figure 4.3, and it provides M products $a_i \times b_i$, $i = 1, 2, \dots, M$. Summation of these products results in an inner product $a_1 b_1 + \dots + a_M b_M$. This summation is accomplished via the integration property (spatial integration) of the cylindrical lens and is valid because of the mixed binary form of the output, which allows for product summation without the need for carries. The resulting pattern is consequently time-integrated by the N -element, 1-D detector array placed at the back focal plane and at location $f_y = 0$.

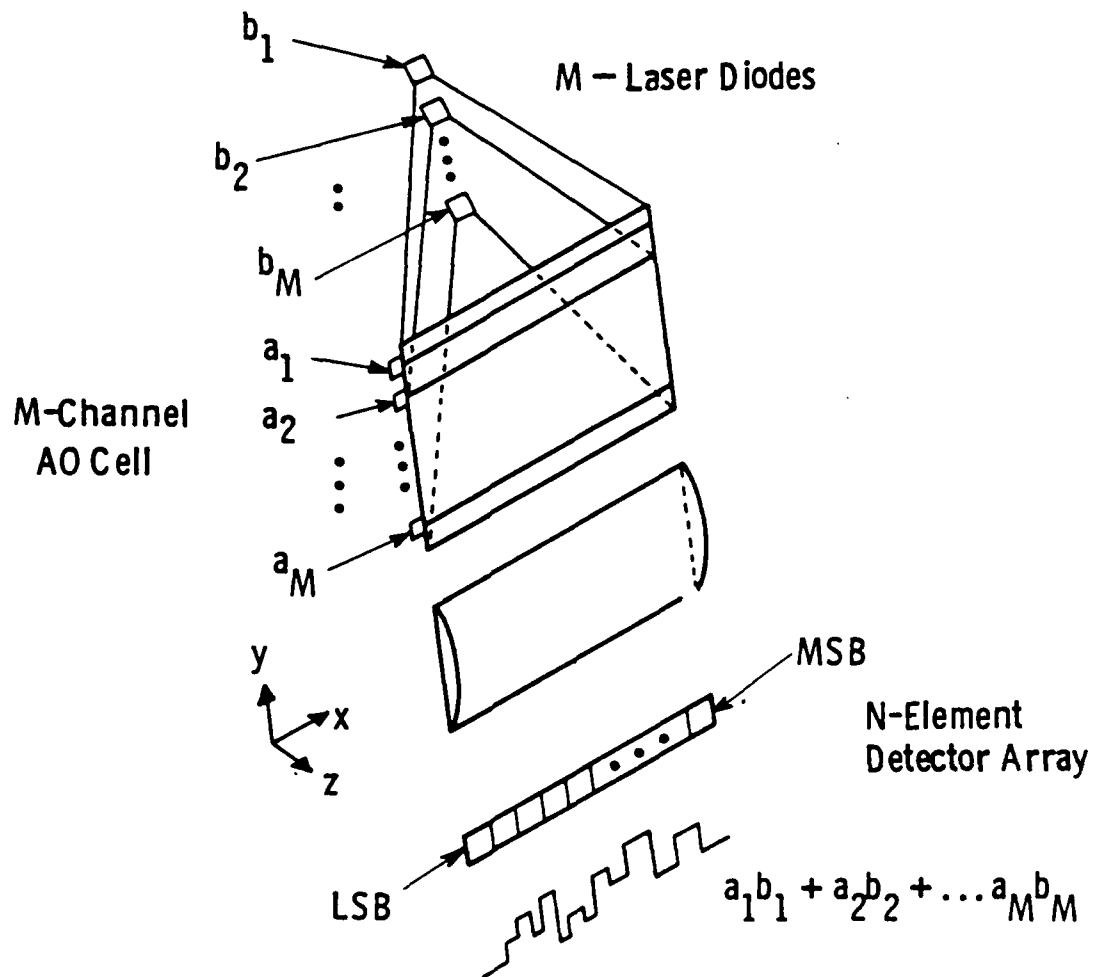


Figure 4.5 Time-Integrating Acousto-Optic Processor for Inner Product Formation

By utilizing the delay properties of a larger aperture M -channel AO cell, in conjunction with an $M \times K$ array of laser diodes, we can extend the system of Figure 4.5 to the systolic processor of Figure 4.6. This system at peak operation can provide, in parallel, K inner products $b_{i1}a_{i1} + \dots + b_{iM}a_{iM}$, $i = 1, 2, \dots, K$ which are formed via the space-integration of the lens and the time-integration of the detector array. Note that in this case the 1-D detector array needs to have $K \times N$ elements. Also note that if the laser diodes illuminate adjacent cell areas (i.e., along x), only $K/2$ inner products are formed during each data cycle. This is because of the requirement that the LSB of the b_i data be followed by $2N$ zeros of total duration $T = NT_B$. We elaborate on these, as well as other, system issues in the following Section.

4.4 DMAC Acousto-Optic Time-Integrating Processor Characteristics

Because of the algorithm used, the system described in Section 4.3 is capable of forming high-accuracy, inner products between bipolar-value vectors. The output accuracy is N bits, which corresponds to a dynamic range of $20 \log(2^N)$. To fully utilize this accuracy, however, one has to minimize possible detection errors, which can be large if we consider the post-processing stage where we effectively weight each mixed binary bit by a power of 2. The key point is to minimize the maximum value accumulated in the detector array, which is $N \times M$. For minimum detection errors⁸ (bit error probability of $< 2.9 \times 10^{-7}$), the detector array should have a dynamic range of at least $10 \times M \times N:1$. This requirement effectively constrains both N and M . Readily available state-of-the-art detectors have a dynamic range of better than 35 dB which, in principle, allows systems with $N = M = 16$. The throughput rate of such a system highly depends on the data loading rate. Currently available AO cells and laser diodes allow for bit widths down to 2 nsec. This means that for $N = 16$, we need at least $2 \times 16 \times 2 \text{ nsec} = 64 \text{ ns}$ for product formation, after loading the data in the AO cell. Consequently a system with $N = K = M = 16$ has a throughput rate of $2 \times 10^9 \text{ M-A/sec}$. The A/D requirements

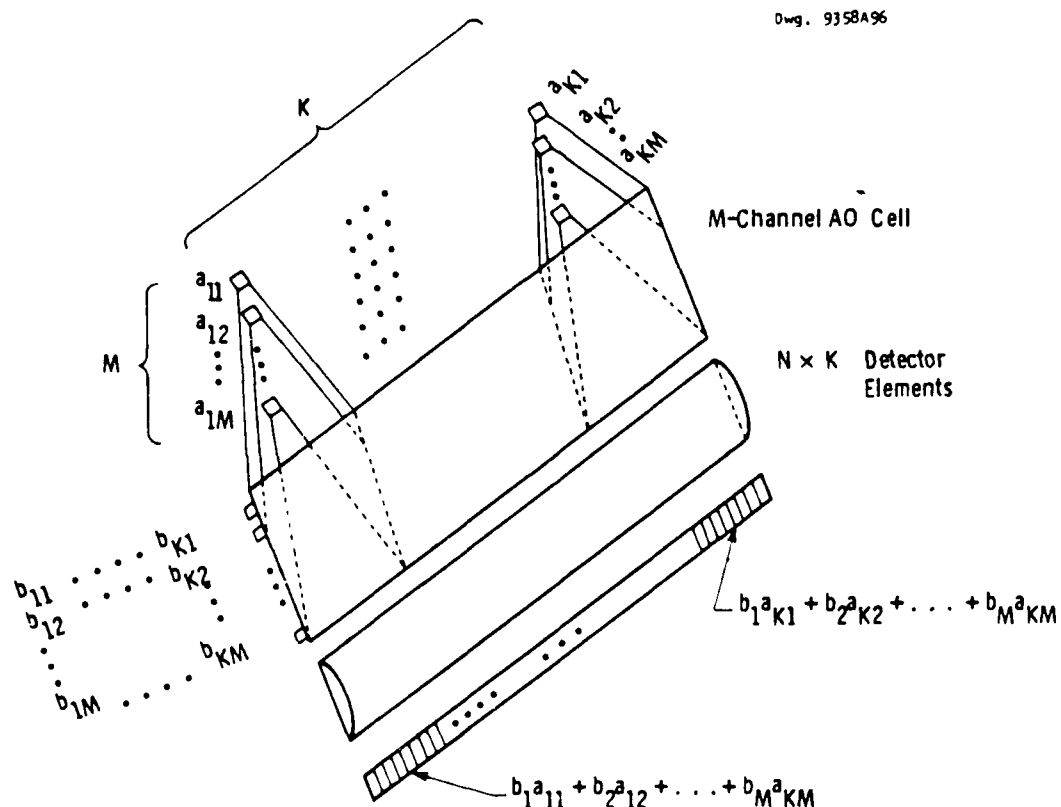


Figure 4.6 Time-Integrating Acousto-Optic Binary Systolic Processor

depend on the number of bits we are reading out which in turn depends on the location of the detector element we are reading out. For example, the maximum possible value for the sign-bit element is $M \times N$, for the MSB is $M \times (N-1)$, for the MSB-1 is $M \times (N-2)$, etc. To read these values, the A/D's need to have $\log_2(M \times N)$, $\log_2(M \times (N-1))$, ..., etc., bits, respectively. A simple analysis shows that, in order to read out 8 inner products, we need 64 A/D's at 8 bits, 32 A/D's at 7 bits, 16 A/D's at 6 bits, ..., etc. These A/D's need to operate at a minimum input frequency of ~ 16 MHz.

Other system issues we need to consider are: (1) laser diode collimation and (2) output detectors. The former depends on both N and the sound speed in the AO crystal. For an efficient system a GaP AO cell should be used because of its good diffraction efficiency ($> 30\%$ /RF watt) and wide bandwidth (> 500 MHz 3-dB bandwidth). In this material the sound speed is $v_s = 6.3 \times 10^6$ mm/sec which implies that, for $N = 16$ and bit + zero width of $T_B = 4$ nsec, the total time-duration of a binary word is $T_W = 16 \times 4$ nsec = 64 nsec. This corresponds to a distance of ~ 0.4 mm over which a single laser diode needs to be collimated. An additional constraint is the maximum allowable light cross-talk between adjacent diodes. To avoid detection errors the cross-talk should be down by the same order as the dynamic range of the detector; e.g., for $N = K = 16$, $M = 16$, it should be at least -34 dB. These requirements can be efficiently met via the use of a fiber-optic fan-out⁸ in conjunction with an array of miniature graded-index collimation lenses. Note that at any instant, every other laser diode is off. In principle, one can take advantage of this by turning off the corresponding detector arrays. This would guarantee absence of cross-talk-related detection errors. To achieve this, one needs detectors that can operate at the binary word switching frequency. For $N = 16$ and $T_B = 4$ nsec the switching frequency is $f_s = 1/(16 \times 4 \text{ nsec}) \sim 16$ MHz, which implies that the detectors should have an integration time of 64 nsec or a clock frequency of 16 MHz. Parallel-readout detector arrays

with the above characteristics can, in principle, be made with current technology. An alternative solution involves a fiber-optic fan-out in conjunction with separate detector elements. This solution guarantees not only the proper integration time but also minimal detector cross-talk which, in turn, minimizes unwanted detection errors.

4.5 BPAM Acousto-Optic Processor

In this section we describe one possible architecture which we have developed for implementing BPAM. Consider the AO processor of Figure 4.7. Light from 4 different laser diodes at wavelengths $\lambda_0, \lambda_1, \lambda_2, \lambda_3$ is multiplexed in a fiber using conventional fiber-optic techniques. The light level from each diode has a value proportional to A_i , $i = 0, 1, 2, 3$. Thus light at λ_0 has a value proportional to A_0 , at λ_1 proportional to A_1 , etc. Thus, if a 4 bit binary representation is used, e.g., 1011, then light will be "on" from lasers at λ_0, λ_2 and λ_3 , and "off" from the laser at λ_1 . The fiber output is then collimated (via lens L_1) and expanded (via lenses L_2, L_3) along the y-dimension. Along the x-dimension the light is focused. The so-created "pencil" beam illuminates a 4-channel Acousto-Optic cell. Each of the 4 channels of the AO cell has a value proportional to B_i . Thus the bottom channel gets the B_0 value, the next channel the B_1 , etc. If B_0, B_1, B_2, B_3 is binary, e.g., 0110, then we will have the two middle channels "on" and the bottom and top channels "off". The AO device is followed by a prism and a cylindrical lens. The prism/lens set accomplishes the following: (1) wavelength demultiplexing and (2) focusing of the 16 spots. Thus, the back focal plane of L_4 we obtain 16 spots in a 4x4 format. Assuming that $\lambda_0 > \lambda_1 > \lambda_2 > \lambda_3$ then the values in the set of spots are:

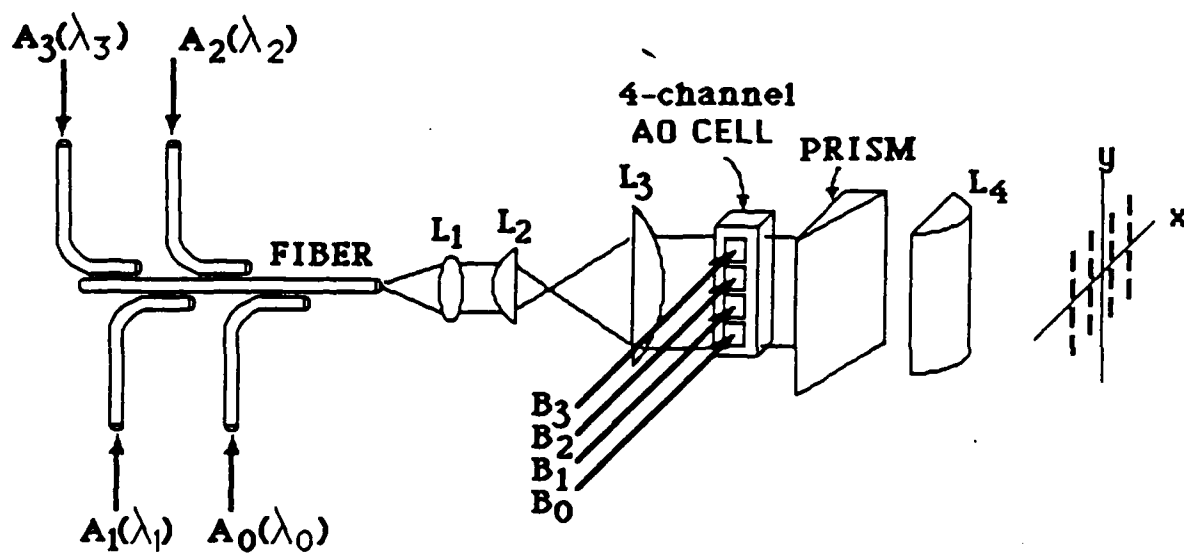


Figure 4.7 BPAM Implementation with AO Cell and λ -Multiplexing

Output Spots	y	A_3B_3	A_2B_3	A_1B_3	A_0B_3
		A_3B_2	A_2B_2	A_1B_2	A_0B_2
	<hr/>				
	x	A_3B_1	A_2B_1	A_1B_1	A_0B_1
		A_3B_0	A_2B_0	A_1B_0	A_0B_0

Close inspection of the values of these spots reveals that we have formed in parallel all A_iB_j , $i, j = 0, 1, 2, 3$, products necessary for the formation of the 7 bits (see Eq. (3.7)). Next we need to add these products properly in order to obtain the 7 convolution points P_0-P_6 .

The additions can be accomplished in a variety of ways:

- (a) The use of detectors with specific area shapes (e.g., see Figure 4.8). The shapes are such that the proper products are added instantly. This is shown in detail in Figure 4.8. Comparison of the results of Figure 4.8 and those of the previous Section shows that we indeed obtain the correct convolution points.
- (b) The use of 16 fibers each of which collects light from a particular spot. The proper fibers are combined onto a single element detector which adds the light leaving the fibers' output. In this case we need 16 fibers and 7 detectors.
- (c) The use of a cylindrical lens which is set at 45° with respect to the y-dimension. This orientation of the lens results in a collapse of the data, which are then read-out by 7 detectors.

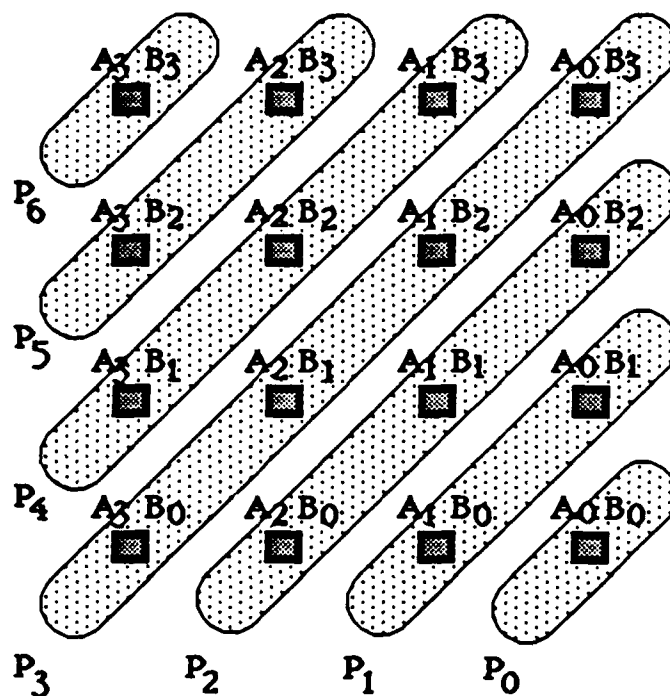


Figure 4.8 Product Summation Via Detector Area Shape

Note that all three techniques are simple and equivalent. The choice depends on the particular design, processor size, etc. Finally, note that the 4-channel AO device can be replaced by a single-channel device in conjunction with frequency multiplexing. In this case, data B_0, B_1, B_2, B_3 will be at frequencies f_0, f_1, f_2, f_3 . Note however that the AO device linearity requirements are increased in order to avoid spurious effects due to the presence of 4 RF frequencies.

4.6 BPAM AO Processor Characteristics

The BPAM AO processor allows us to perform a digital multiplication every clock cycle (not every $2N-1$ cycles as in DMAC). This technique fully utilizes the inherent speed of optics. In principle, the system can multiply (and if we desire accumulate) binary numbers as fast as the lasers or AO cells can be switched. State-of-the-art lasers can be switched every 0.3 nsec (this translates to a throughput rate of 3 GOPS). For a practical scenario, however, the limiting factor will be the speed with which the AO cell input data can be provided. With currently available AO technology, a miniaturized unit of the processor of Figure 4.7 could be operated at 250 MHz (see Section 4.7). In this case the system's throughput rate is 0.25×10^9 M-A/sec.

The number of wavelengths λ_i is equal to the number of bits in the input. To avoid an extensive number of λ_i (and AO channels) we use a base system higher than 2. A good choice is base 4 with 4 digits. Then we obtain 8 input binary bits and 16 output binary bits. In this case we need 4 wavelengths, 4 AO channels, 7 detectors, and 7 A/D converters. Note that in this case we also need 4 input light levels; that is, 0, 1, 2 and 3.

To avoid extensive A/D requirements, we use time integration for summation of different number products (i.e., creation of inner products or vector-vector multiplication). The time integration can be

accomplished if each detector is followed by a charge integrator. For clock speeds of 250 MHz, and for summation of 7 products, the A/D requirements are 7 A/D's with 8 bits each at 29 MHz.⁹

4.7 BPAM AO Response Analysis

In this Section we address the speed of the BPAM AO processor and, in particular, assess the ultimate speed limit of a BPAM-based AO processor. There are three key components in the system of Figure 4.7: (1) laser diodes, (2) AO cell and (3) output detectors and A/D's. The speed limit of the first component is usually in the range of a few GHz. For example, the LD53-OMF laser diode manufactured by ORTEL Corporation has a modulation bandwidth of 8 GHz. This translates to ON/OFF pulses of 0.167 nsec which in turn translates to data rates of 1 bit/(2 x 0.167 nsec) \approx 3 Gb/s, assuming RZ data format. Thus, this component allows throughput rates of the order of 3×10^9 M-A/sec. The speed limit of the last component (detectors and A/D's) depends on whether we use time-integration (i.e., detection of inner products instead of single products). For most applications of interest, we require vector-matrix multiplication or matrix-matrix multiplication. This implies that the desired outputs are inner products, which in turn implies that time-integration is preferable. For these applications, the A/D speed requirement is about 1/time for inner product formation. Assuming that we are forming inner products that consist of 32 or more products, and we allow 0.167 nsec per product and 0.167 nsec per zero (each product is followed by a zero), we find that the time required for the formation of an inner product is $32 \times (0.167 \text{ nsec} + 0.167 \text{ nsec}) \approx 10 \text{ nsec}$. Thus the A/D speed requirement does not exceed 100 MHz which is within the capabilities of the present A/D technology.

Let us now examine the second component, namely, the AO cell. It can be shown¹⁰ that the modulation bandwidth Δf associated with a response falloff (in dB) of β , is

$$\Delta f = \frac{0.7 \sqrt{\beta} U_s}{n_A W_o} , \quad (4.5)$$

where U_s is the speed of sound in the AO crystal, n_A is the refractive index of the AO material, and W_o is the diameter of the input laser beam at the e^{-2} points. Let us assume that $\beta = 3\text{dB}$, $U_s \approx 6.32 \times 10^3 \text{ m/sec}$, and $n_A = 3.31$, which corresponds to a scenario with a GaP AO cell (which is one of the "fastest" AO cell materials). In this case Equation 4.5 becomes:

$$\Delta f = \frac{2.31 \times 10^3 \text{ m/sec}}{W_o} \quad (4.6)$$

To calculate Δf and, thus, the AO cell's "speed of response," we need to know the laser beam diameter W_o . The diameter W_o of the focused laser beam is given by:

$$W_o = S \cdot f_L , \quad (4.7)$$

where S is the angular spot radius and f_L is the focal length of the lens used to focus the laser beam. The angular spot radius S is given by

$$S = \frac{1}{2} K(N_o) (D/f_L)^3 + \frac{1.22 \lambda}{n D} \quad (4.8)$$

where D is the beam diameter incident on the lens, n is the index of refraction in air and $K(N_o)$ is an explicit function of index ratio $N_o = n'/n$ (n' is the lens material index of refraction) and lens shape. In Equation 4.8, the first term represents the contribution of spherical aberrations to the beam size W_o . The second term represents the

contribution of diffraction effects to the beam size W_o . Clearly, we wish to optimize the value of D to give the minimum value of S . This optimization has been carried out numerically as a function of f_L for the case of a plano-spherical lens which has the smallest $K(N_o)$ given by

$$K(N_o) = \frac{1}{32(N_o - 1)^2} (N_o^2 - 2N_o + 2/N_o). \quad (4.9)$$

As fixed data we use: $\lambda = 780 \mu\text{m}$ (corresponding to a high-speed AlGaAs laser diode), $N_o = 1.51106$ (corresponding to BK-7 lens material). The results of the calculations are given in Table 4.1 which shows the spot size W_o as a function of focal length f_L together with the optimum D, D_{opt} , for minimization of Eq. 4.8.

Table 4.1 W_o as a Function of f_L

f_L (mm)	D_{opt} (mm)	W_o (μm)
1.00	0.31	4.14
2.00	0.52	4.92
3.00	0.70	5.44
4.00	0.87	5.85
5.00	1.03	6.18
6.00	1.19	6.47
7.00	1.33	6.73
8.00	1.47	6.95
9.00	1.61	7.16
10.00	1.74	7.35

The smallest possible focal length that we can hope to use with a state-of-the-art GaP AO cell is 2 mm, dictated by the crystal width

necessary to provide uniformity in the lapping process of the A0 cell's transducer. From Table 4.1 we see that for $f_L = 2 \text{ mm}$, the diameter W_o of the focused spot is $4.92 \text{ } \mu\text{m}$. Substituting for $W_o = 4.92 \text{ } \mu\text{m}$ in Eq. 4.6, we find that the modulation bandwidth Δf of the A0 cell cannot exceed 470 MHz. Thus, each bit in the A0 cell will be represented by a pulse that has a width of $\approx 2 \text{ nsec}$. This corresponds to a data rate (assuming RZ data format) of

$$\text{Data rate} = 1/(2 + 2 \text{ nsec}) \approx 250 \text{ Mb/s} \quad (4.10)$$

In conclusion, we find that the component that sets the limit, in the BPAM A0 system, is the A0 cell. We also find that, in the best case, the data rate in the A0 cell is $\approx 250 \text{ Mb/s}$ which corresponds to a throughput rate for the BPAM A0 system of $250 \times 10^6 \text{ M-A/sec}$.

5. CIRCULARLY POLARIZING SAMPLING TECHNIQUE FOR COMPLEX MATRIX OPERATION

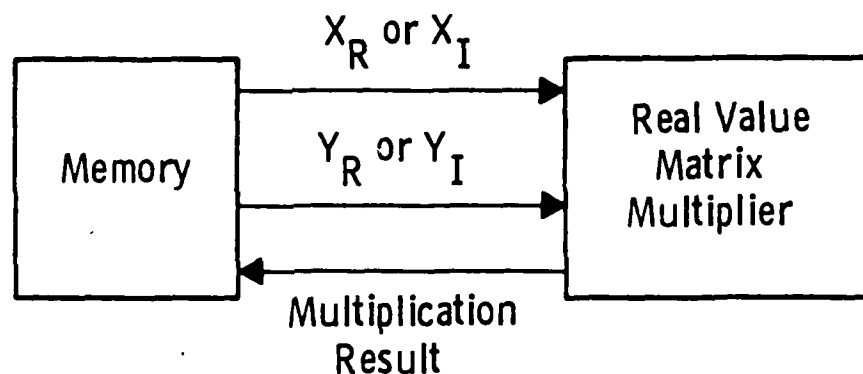
In the previous Section we have described Acousto-Optic architectures which can be used for performing matrix multiplication operations. Matrix multiplication is the most essential and major operation for our applications of interest. So far, the elements in those matrices have been treated as real numbers. In reality, however, the elements are generally complex numbers. This is an unavoidable situation, especially when the original signals are acquired from heterodyning processes that yield the quadrature pair. Therefore it is very important to provide an efficient method of handling complex numbers in order to achieve efficient matrix multiplication operation. There are two conventional techniques to accommodate complex numbers; both have serious problems. The first method is a well-known software solution that realizes the complex multiplication by first decomposing the operation into four independent multiplications of real matrices (Figure 5.1) to form real x real, real x imaginary, imaginary x real, and imaginary x imaginary terms, and later synthesizes the real part and the imaginary part of the final product by summing the square terms and the cross terms, respectively. This method allows us to utilize a matrix multiplier for real numbers without any modification in the hardware. The disadvantage of this method is the slow speed which results from the necessity of reading the buffer memory (which contains the real and imaginary parts) four times. Thus, it takes four times longer than the multiplication of real matrices.

The second method is a hardware solution. It requires a major modification of the multiplier cells to accommodate complex numbers (Figure 5.2). Each cell must contain four multipliers and two adders and must compute the four terms in parallel. The problem with this method is the tremendous complexity of its hardware. It requires that the size of each cell be increased by a factor of four and that the

Matrix Multiplication : $Z = XY$

$$Z_R = \underbrace{X_R Y_R}_I - \underbrace{X_I Y_I}_{II}$$

$$Z_I = j \left(\underbrace{X_R Y_I}_{III} + \underbrace{X_I Y_R}_{IV} \right)$$



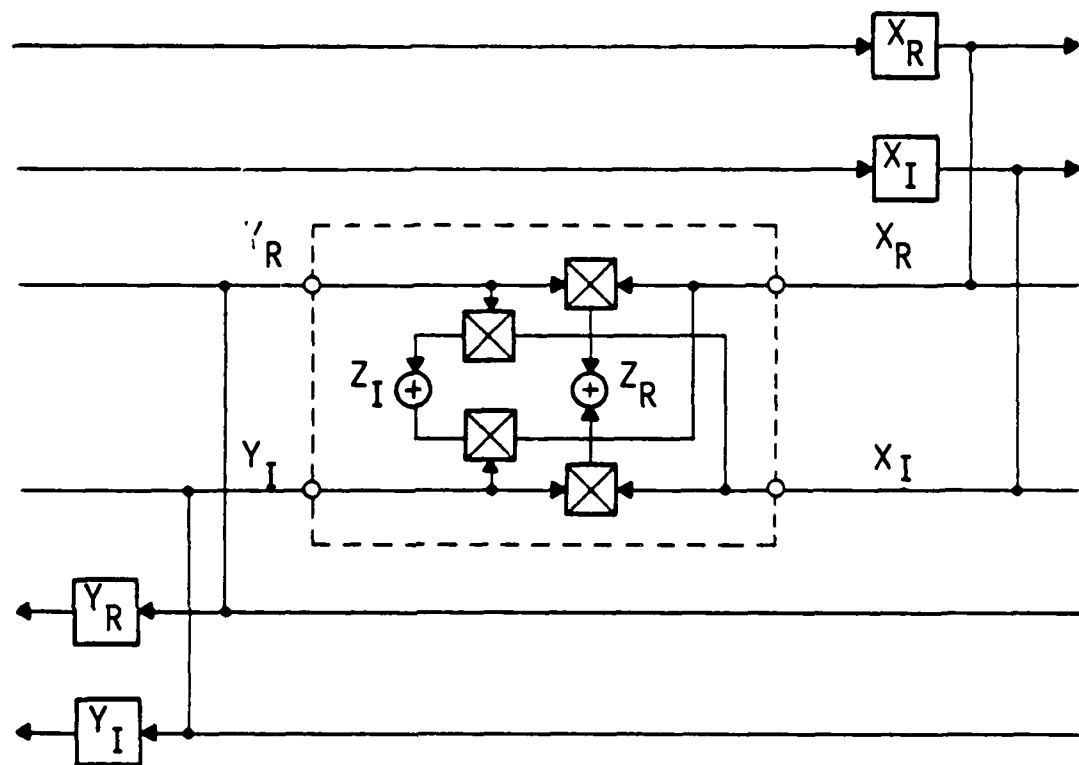
Problem

Slow Speed

The Correlator is Required to Read the Memory 4 Times.
Then it Needs Two More Adding Operation Steps to Obtain
the Complex Pair.

Figure 5.1 Multiplication of Complex-Valued Matrices Using Real-Valued Matrix Multiplier

Structure to Accommodate Complex Numbers



Problem

Hardware Complexity
 Input Data Flow Needs Double Channel
 Each Cell is Required to Have 4 Multipliers and 2 Accumulators

Figure 5.2 Complex-Valued Multiplier Cell

number of the input ports be doubled. It is very difficult to realize such hardware, especially in the AO architectures.

In the following paragraphs we present a unique solution that is able to maintain relatively high speed and also hardware simplicity at the same time.

A complex signal carries more information than a real signal. Therefore, the manipulation of such signals becomes more computation intensive. However, we can improve the data handling significantly by arranging the input data in the most efficient way. One major complexity in handling the complex matrix multiplication is the duality: every sampled data point comes in the form of a pair, composed of the real part time series and the imaginary part time series. Our approach solves the complexity problem by forming a composite but single-time series that is capable of representing both the real and imaginary parts. Thus, the elements of the matrix have a single number instead of a real and imaginary pair. This arrangement can simplify the matrix multiplication operation very significantly. The approach consists of two steps: (1) a special sampling process and (2) the use of such samples in the matrix multiplication.

5.1 Circularly Polarizing (CP) Sampling

A typical data acquisition process is depicted in Figure 5.3. First, the original signal is received in the antenna and mixed with the local oscillator of the target frequency by using an ordinary heterodyne process that yields an analog quadrature pair representing the real part and the imaginary part. In the conventional technique those signals are digitized by A/D converters which are strobed by the same clock to generate a digital quadrature pair in every sampling period, t_s . In the present CP sampling scheme, the two A/D converters are strobed by clocks having the same frequency but phase shifted by 180° . Also the sign of the sample in each quadrature is alternated at every sample. The

Dwg. 9359484

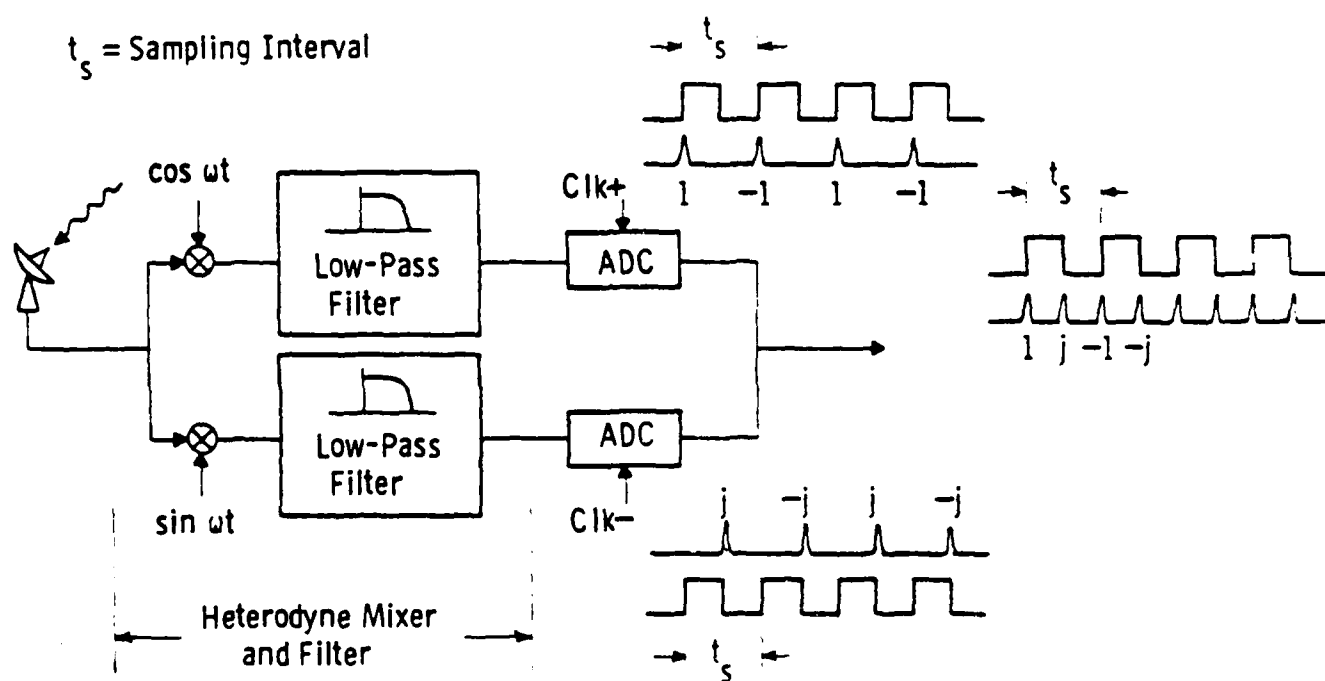


Figure 5.3 Circularly Polarizing Sampling Scheme

digitized results are interlaced to form a single string of sampled data with the interval $(1/2) t_s$. This composite time series has either a progressive quadrature $(1, j, -1, -j, \dots)$ or a regressive quadrature $(1, -j, -1, j, \dots)$, depending on the phase of the polarity alternation. We call the former the right-hand circularly polarizing (RCP) sampling and the other the left-hand circularly polarizing (LCP) sampling, indicating the rotational orientation of the phasor. Appendix A shows that either of the CP sampling schemes is capable of representing both real and imaginary values of the original signal without loss of information and free from the aliasing problem as long as the sampling frequency $(1/t_s)$ is greater than the bandwidth of the original signal. It also shows that RCP and LCP sampled signals are conjugate to each other. The bandwidth requirement is identical to that of the conventional sampling case; namely, the Nyquist criterion. The obvious advantage of this data representation is simplicity. We now show the use of the CP sampled data in performing matrix multiplication.

5.2 Matrix Multiplication Using CP Sampled Data

We first discuss a correlation operation as an example of the simplest case of matrix-vector multiplication. Later we address the more general case.

Consider a correlation operation function $y(\tau)$ for signals $h(t)$ and $x(t)$ where

$$y(\tau) = \int h^*(t-\tau) x(t) dt \quad (5.1)$$

and $h(t)$, $x(t)$, and $y(\tau)$ are continuous complex functions and $h^*(t)$ is conjugate to $h(t)$. We can translate the situation to the discrete CP sample domain by

$$y_\tau = \sum_{t=1}^{\infty} h_{t-\tau}^* x_t \quad (5.2)$$

where y , h , and x are CP sample functions and t and τ are integers. Suppose the correlating function h_t has finite duration, say, the duration of four data points. Then we can express the correlation operation in the form of a multiplication between the band matrix H with bandwidth 4 and the vector X (see Figure 5.4). Now the input function x_t is a CP sampled function carrying the sequential quadrature information $(1, j, -1, -j, \dots)$, and the correlating function $h^*(t)$ is the conjugate of h_t . Therefore, it has the corresponding LCP quadrature sequence $(1, -j, -1, j, \dots)$. The first element of the output vector y_1 is the result of the inner product between the first row of the matrix H and the vector X , or

$$\text{1st element: } (h_1)(X_1) + (-jh_2)(jX_2) + (-h_3)(-X_3) + (jh_4)(-jX_4). \quad (5.3)$$

Note that the phase of all the product terms turns out to be 0° . Therefore, the first element can be expressed as:

$$y_1 = h_1X_1 + h_2X_2 + h_3X_3 + h_4X_4 \quad (5.4)$$

It is clear that this inner product calculation requires only the multiply/add capability of real numbers. The second element in the output vector is calculated similarly.

$$\text{2nd element: } (h_1)(jX_2) + (-jh_2)(-X_3) + (-h_3)(-jX_4) + (jh_4)(X_5) \quad (5.5)$$

In this case, the phase of the entire term is a constant 90° . Thus, it is appropriate to represent the second element as:

$$jy_2 = j(h_1X_2 + h_2X_3 + h_3X_4 + h_4X_5) \quad (5.6)$$

$$\underbrace{\begin{bmatrix}
 h_1 & -jh_2 & -h_3 & jh_4 & & & & \\
 & h_1 & -jh_2 & -h_3 & jh_4 & & & \\
 & & h_1 & -jh_2 & -h_3 & jh_4 & & \\
 & & & h_1 & -jh_2 & -h_3 & jh_4 & \\
 & & & & \cdot & \cdot & \cdot & \cdot \\
 & 0 & & & \cdot & \cdot & \cdot & \cdot
 \end{bmatrix}}_H \begin{bmatrix} x_1 \\ jx_2 \\ -x_3 \\ -jx_4 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} y_1 \\ jy_2 \\ -y_3 \\ -jy_4 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

X Y

NOTE: The phase of the product function between a row of H matrix and the X column vector is always constant; thus real valued multiply-add capability is sufficient to calculate the output vector Y.

Figure 5.4 Correlation Operation Using CP Sampling Approach

so that the value y_2 still remains real. In a similar way, we can obtain the third and fourth elements:

$$-y_3 = -(h_1X_3 + h_2X_4 + h_3X_5 + h_4X_6) \quad (5.7)$$

$$-jy_4 = -j(h_1X_4 + h_2X_5 + h_3X_6 + h_4X_7) \quad (5.8)$$

It is clear that the rest of the elements can be obtained in the same way. In any of these inner product calculations, none of the complex-valued multiplication capability is needed, owing to the CP sampling. The output vector is again in the form of CP samples.

Thus, we see that once we represent the input complex signal in CP sample form, the output is also in CP sample form, and we can carry out the entire operation in the CP sample domain. The method treats all the numbers as real numbers and the quadrature information is coded in the data position itself.

In the case of the more general shift variant system, the matrix is not necessarily banded and there is no repetition of the same series from row to row as is the case in correlation or convolution operation. However, the matrix-vector multiplication will still be accomplished in the same manner as long as the data position represents the proper quadrature. An example of positional coding of phase is shown in Figure 5.5. An interesting characteristic of the matrix is that the quadrature is progressing along the two-dimensional matrix and constant along the diagonal orientation. As long as this quadrature based pattern is used, we can perform multiplication of any matrices, using only real numbers.

In summary, we believe that this new CP sampling approach solves, to a great extent, the problems of the matrix-vector multiplication for

$$\begin{bmatrix} 1 & -j & -1 & j & 1 & -j & -1 & j \\ j & 1 & -j & -1 & j & 1 & -j & -1 \\ -1 & j & 1 & -j & -1 & j & 1 & -j \\ -j & -1 & j & 1 & -j & -1 & j & 1 \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ j & 1 & -j & -1 & j & 1 & -j & -1 \\ -1 & j & 1 & -j & -1 & j & 1 & -j \\ -j & -1 & j & 1 & -j & -1 & j & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ j \\ -1 \\ -j \\ 1 \\ j \\ -1 \\ -j \end{bmatrix} = \begin{bmatrix} 1 \\ j \\ -1 \\ -j \\ 1 \\ j \\ -1 \\ -j \end{bmatrix}$$

NOTE: The figure shows the positional coding of quadratures and amplitude of each element is arbitrarily set to 1.

Figure 5.5 Positional Quadrature Representation in Matrix-Vector Multiplication

complex numbers in a very systematic way. It is important to note that the total operation is carried out in the CP sample domain, and that it gives us the freedom to cascade such operations, or the freedom to iterate the operation by feeding back the result of one multiplication to the next without an intermediate domain conversion process.

6. PERFORMANCE OF HIGH ACCURACY ACOUSTO-OPTIC PROCESSORS

In this Section we consider the performance of the high accuracy AO processors which we have developed (Section 4). It is of interest to compare these systems with typical, state-of-the-art electronic devices. In describing system performance, one measure that is usually employed is the throughput rate (TR). However, when comparing different systems, it is also of interest to examine the efficiency of the system (SE), defined as the throughput rate per unit power, along with the net multiplication speed (MS). The efficiency measure is important because it demonstrates the power consumption necessary for a given TR. The multiplication speed measure is useful because in certain applications high MS rather than high TR is important.

The comparison analysis is done by calculating the SE and MS figures for various families of state-of-the-art electronic multiplier accumulators as well as for typical space-integrating and/or time-integrating AO processors that employ DMAC or BPAM. These calculations are used as the basis for a simple, first-order, comparison which gives a clear picture of the computational competitiveness of AO processors.

6.1 Performance of Electronic Multipliers

Current, state-of-the-art, electronic competition comes from three families of electronic integrated circuits: (a) CMOS, (b) high-speed, Silicon-based, VLSI, and (c) GaAs.

In the first category we have a variety of commercially available multiplier/accumulator chips such as: (a) the Toshiba T6354 16 input bit, 32 output bit (16/32 bit) chip and (b) the Logic Devices LMA 1009-1 16/32 bit device. The first device has an MS of 10 MHz and a power consumption of 100 mW. This corresponds to an SE of 100×10^6 M-A/sec.W. The second device has an MS of 15 MHz with a power

consumption of 125 mW. This corresponds to an SE of 120×10^6 M-A/sec.W.

In the second category we have various high-speed VLSI devices which are usually custom made for specific signal processing applications and can contain a large number of multiplier/accumulator units. Westinghouse's typical high-speed 8/16 bit VLSI is capable of an MS of 30 MHz while consuming about 250 mW. This corresponds to an SE of 120×10^5 M-A/sec.W.

Finally, in the third category, we have a number of GaAs LSI devices an example of which is the Rockwell¹¹ 8/16 bit multiplier. This device forms the 16 bit product in 5.25 nsec which corresponds to an MS of 190 MHz. Power dissipation is about 1.4W. Thus, the device's SE is 135×10^6 M-A/sec.W.

The results of the above calculations are compiled in Table 6.1. From this table we see that although the MS varies from family to family (10,30,190 MHz, respectively) the SE remains about the same ($\sim 120 \times 10^6$ M-A/sec.W).

6.2 Performance of DMAC Based AO Processors

We begin the analysis of AO processors by considering DMAC based systems. We first examine the performance of the space-integrating single detector AO system of Section 4.1. We choose to consider this system first because it represents a typical example of the ability of optics not only to multiply but also to compress (i.e., add) data. Let us assume that the input accuracy is 8 bits. In this case and for $M = 32$ the maximum value of the output convolution is $8 \times 32 = 256$, which requires an 8-bit A/D. State-of-the-art (chip level) 8-bit A/D's operate at 30 MHz and consume 120 mW of power⁹. Use of 3 such A/D's in conjunction with an electronic data deflection scheme allows a

conversion rate of 90 MHz at a power consumption of about 500 mW. Such a rate allows data periods of about 11 nsec. The convolution operation is completed in $2N-1 = 15$ cycles and thus the MS is 6 MHz and the TR is 192×10^6 M-A/sec. For the purposes of this analysis we assume that the bulk part of the power consumption comes from the laser and the A/D's. A detailed analysis shows that the laser consumption is about 5 W. Thus, the total power consumption is 5.5 W and the SE is 35×10^6 M-A/sec.W. This is a rather poor figure and is partially due to the low MS figure. To improve this figure let us assume that 15 A/D's are used so that effective conversion rates of 450 MHz are achievable. In this case we can use clock frequencies of 900 MHz and have an MS of 30 MHz, a TR of 960×10^6 M-A/sec and an SE of 128×10^6 M-A/sec.W. Comparison of these figures with those of the electronic cases, however, shows that the AO system does not offer any significant SE or MS advantage.

We now examine a different AO system architecture, specifically an array, non-compressive processor. This system is similar to the one of Figure 4.2, but it uses a cylindrical lens (with power along x) in conjunction with M detectors for the individual computation of the M products. In this case, for an 8/16 bit system, the maximum convolution value is 8 which requires a 3-bit A/D. We assume that a set of 8-level comparators will be used because of the relatively low resolution required. For 8-level comparison, 3-dual level comparators are needed along with some dedicated logic. A typical example of a high-speed comparator is the Advance Micro Devices AM6687 which allows us to build an 8-level comparator circuit which operates at about 300 MHz and has a power consumption of about 800 mW. This allows a clock frequency of 600 MHz which translates to an MS figure of 20 MHz. Thus for $M = 32$, the system's TR is 640×10^6 M-A/sec. To calculate the SE figure we need to calculate the laser power. If we use one laser diode per AO channel we find that 5 mW, 20% efficient, laser diodes are sufficient.

Table 6.1

TYPICAL PERFORMANCE OF ELECTRONIC MULTIPLIERS

FAMILY	TYPE	I/O (BITS)	MS (MHz)	POWER (W)	SE (M-A/sec W)
CMOS	Toshiba T6354	16/32	10	0.100	100×10^6
CMOS	Logic D. LMA1009-1	16/32	15	0.125	120×10^6
High Speed VLSI	Westinghouse	8/16	30	0.250	120×10^6
GaAs	Rockwell	8/16	190	1.4	135×10^6

In this case, the total laser diode power consumption is $25 \times 32 \text{ mW} = 800 \text{ mW}$ and the total power consumption 26 W . This gives an SE figure of $23 \times 10^6 \text{ M-A/sec.W}$. Comparison of the system's performance figures with those of the electronic counterparts shows that the AO system, once again, does not offer any significant advantage.

We now examine the time-space integrating systolic AO processor of Section 4.3.

For minimum detection errors, and with state-of-the-art components, a 8/16 bit system with $N = K = 16$ and $M = 8$ is realistic. The TR of such a system depends on the data loading time. Currently available AO cells allow for bit widths of 2 nsec. Thus, each multiplication requires $2 \times 8 \times (2 + 2) \text{ nsec} = 64 \text{ nsec}$. This corresponds to an MS of 15.6 MHz and a TR of $8 \times 8 \times 15.6 \times 10^6 \text{ M-A/sec}$ or 10^9 M-A/sec . To calculate the power consumption we take into account the laser diodes and the A/D's only. An 8×16 laser diode array (with 5 mW, 20% efficient diodes) requires an average power of 1.6 W. The power consumption of the A/D's depends on the number of A/D's used and the operating frequency. For full use of the system's MS we need to use a combination of serial and parallel detector read out. With such a scheme we need 64 7-bit A/D's, 32 6-bit A/D's, 16 5-bit A/D's, etc., that operate at 16 MHz. If we use the 30 MHz 120 mW 8-bit A/D's we can replace the 64 7-bit A/D's (i.e., two elements per A/D). The total power consumption of these devices is 3.84 W. This figure represents $\sim 70\%$ of the total A/D power consumption. Thus, for our purposes, the total power consumption is $\sim 7.1 \text{ W}$ and gives an SE of $141 \times 10^6 \text{ M-A/sec.W}$. Comparing these figures with those of the electronic competition (Table 6.1) we find that the AO system does not offer any significant performance advantage.

In conclusion, we see that the DMAC-based architectures do not offer any significant performance advantages (Table 6.2) when compared with

Table 6.2

TYPICAL PERFORMANCE OF DMAC AO SYSTEMS

TYPE	I/O (BITS)	MS (MHz)	POWER (W)	SEC (M-A/sec W)
SPACE INTEGRATING (1 Detector, 3 A/D's)	8/16	6	5.5	35 x 10 ⁶
SPACE INTEGRATING (1 Detector, 15 A/D's)	8/16	30	6.8	128 x 10 ⁶
SPACE INTEGRATING (32 Detectors, 32 Comparators)	8/16	20	26	22.7 x 10 ⁶
TIME/SPACE INTEGRATING	8/16	15.6	7.1	141 x 10 ⁶

their electronic counterparts (Table 6.1). Similar performance is characteristic of other architectures that have appeared in the open literature but for reasons of space have not been included in this Section. There are two main factors which limit this performance. First, the algorithm itself takes $2N-1$ cycles to complete the convolution. This results in the requirement of at least one bit-serial propagation and thereby substantially reduces the TR (which in principle can be high). Second, optics performs only part of the full, high-accuracy multiplication, namely, the convolution, and subsequently requires the "help" of power-consuming electronics (e.g., A/D's) to complete the operation. This results in an increased power consumption and a decreased SE.

Based on the above observations we conclude that in order to improve the MS and SE figures, we need to decrease the time required for completion of the convolution and eliminate the A/D's.

6.3 Performance of BPAM Based AO Processors

Since the BPAM operation requires one clock cycle, the system TR is equal to the speed with which we can address the AO cell. In Section 4.7 we showed that at best the AO device can be operated at about 250 MHz which corresponds to a TR of 250×10^6 M-A/sec. and an MS of 250 MHz. The power consumption of the system depends on: (a) the number of bits in the input and (b) the number of A/D's used. To avoid an extensive number of wavelengths we need to use a base system higher than 2. If we use base 4 with 4 digits then we have a processor of 8/16 bits. In this case we need 4 laser diodes, 7 detectors and 7 A/D's. If we use the 8-bit 30 MHz 120 mW A/D's, then we can read out the output at a rate of 29 MHz. Each such output will correspond to an inner product which is the sum of 8 number products (the summation is performed in time, at the detectors). In this case the A/D power consumption is $7 \times 120 \text{ mW} = 840 \text{ mW}$. Adding to this figure the power

consumption of the laser diodes ($4 \times 25 \text{ mW} = 100 \text{ mW}$) we find that the power consumption of the system is 940 mW. This corresponds to an SE of $266 \times 10^6 \text{ M-A/sec.W.}$

Comparing the MS and SE figures of the BPAM-based AO system with those of the DMAC based AO systems (Table 6.2), we find that the MS figure of the BPAM system is higher by about an order of magnitude. This improvement is expected since the time requires for the BPAM is reduced by a factor of $2N-1$. On the other hand, because MS is increased, one might expect SE to increase proportionally. This is not the case, however, because SE is increased only by a factor of 2. This is due to the fact that BPAM requires $2N-1$ output detectors and A/D's for the detection/conversion of a single product. This translates to additional power consumption which partially offsets the MS improvement.

If we now compare the performance of the BPAM-based AO system with that of the electronic competition (Table 6.1), we find that the BPAM system has a MS figure that is higher by about an order of magnitude than that of Silicon-based devices. However, this advantage essentially disappears when compared with GaAs devices. Thus, once again, we find that for all practical purposes the AO systems do not offer any significant performance advantage (e.g., an order of magnitude improvement in SE and/or MS). One of the main reasons for this behavior is the fact that the available algorithms (both DMAC and BPAM) require power-hungry post-detection electronics (i.e., A/D's and comparators) for conversion of a multi-level analog signal to a binary output.

A method of overcoming this problem, is to eliminate the presence of analog signals by employing binary valued $A_i, B_i, i=0, \dots, N-1$ input levels. In this case the various products $A_i B_i$ can take only two values, 0 and 1. This implies that, in the absence of an optically implemented product summation, the N^2 detectors (one for each product) simply detect the presence or absence of light. Subsequently, their

binary outputs are used to directly drive N^2 pulse-counting electronics (e.g., counters) and not A/D's or comparators. Once these devices count M such pulses (for an M -element inner product) they drive properly arranged digital adders that perform the $A_i B_i$ product summations. The outputs from the adders subsequently drive a shift-register/accumulator that performs the final weighting and summation operations. An example of such an electronic arrangement is shown in Figure 6.1 for the case of a 2-bit multiplier.

In optimizing such an architecture, it becomes immediately apparent that, aside from its delay properties, the A0 cell is used as a simple optical switch. In fact, to provide this switch function, the A0 cell is unnecessarily complex because: (a) it requires RF carriers, mixers and amplifiers, (b) it requires a rather complicated optical system, and (c) it requires an 8-channel optical multiplexer (for a 8/16 bit system). A far simpler and faster system can be realized by the use of two sets of N laser diodes in conjunction with $2N^2$ detectors and N^2 AND gates. An example of a possible arrangement of such a system, is shown in Figure 6.2 for a 2-bit multiplier. Other possible arrangements involve the use of fiber-pigtailed lasers in conjunction with the fiber-optic 1:N splitters, or overlapping laser beams with N^2 detectors (located at the cross points) in conjunction with threshold detection, etc. Note that the speed of the optical part of any possible implementation of the system can exceed 3 GHz even assuming the use of state-of-the-art components. Thus, the throughput limiting factor is the AND gates together with the counters that follow them. Finally, a simple analysis shows that a scheme of parallel input counters, which are used in order to obtain the sum of 1's for each convolution point, is preferable to the scheme shown in Figure 6.1. This is the well known Dadda¹² scheme for implementing a fast many-bit ($> 16/32$) digital multiplier.

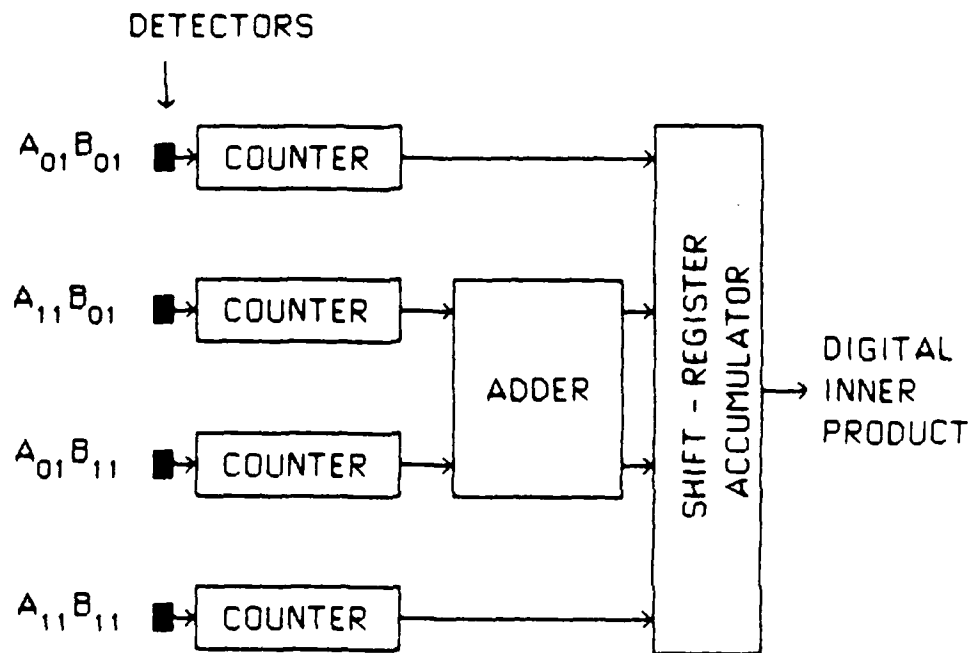


Figure 6.1 Counter Arrangement for a 2-bit Multiplier

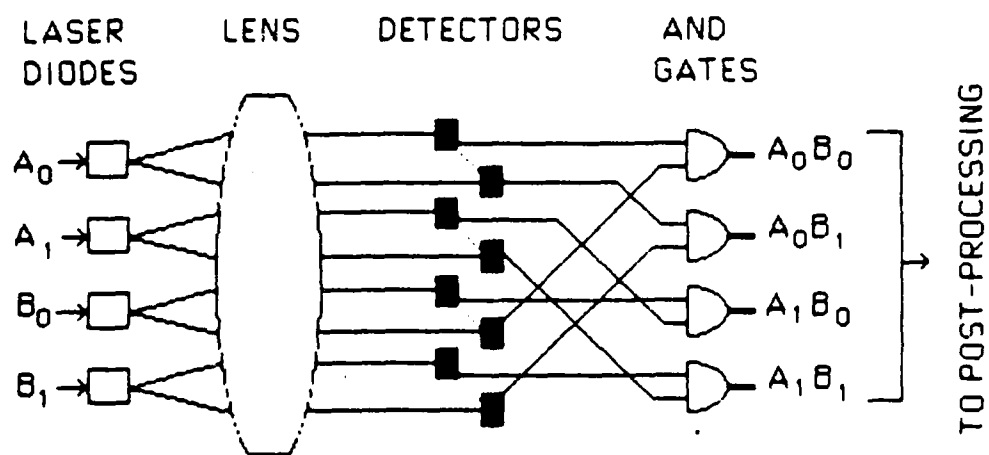


Figure 6.2 Laser-Diode/AND-Gate System for Product Formation

Thus, we have formed a simple multiplier system that does not require any A/D's or comparators and which forms the $A_i B_i$ products in a single clock cycle. The factor limiting the speed of the device is the electronics (~ 200 - 300 MHz with MECL II logic) and not the optics (> 3 GHz). A close look at the device, however, reveals that optics is used only for the high-speed interconnections (between data source and multiplier) and not for the actual product computation which is performed exclusively by dedicated digital electronics. Thus, the efficient implementation of binary-valued BPAM in conjunction with binary detection results in a system where the computational role of optics is practically zero. We discuss such a device in more detail in Section 7.

6.4 Performance Comparison Conclusions

In this Section we have examined the performance of the AO processors from the points of view of system efficiency and multiplication speed. It is found that DMAC-based AO systems do not compare favorably with existing state-of-the-art electronic multipliers. BPAM-based AO systems, although superior to DMAC-based systems, have a performance which is about the same as that of existing GaAs devices. An attempt to use BPAM systems with digital counters, instead of A/D's or comparators, results in a system in which optics is used for the high-speed interconnections but not for computations.

7. OPTICALLY ADDRESSED ELECTRONIC DIGITAL MULTIPLIERS

7.1 Introduction

In the previous sections we have examined techniques that allow Acousto-Optic processors to perform digital-accuracy arithmetic computations. We have shown that when the multiplication speed and system efficiency figures are used as performance measures, the AO systems have no significant advantage over existing state-of-the-art digital electronic multipliers. However, one way in which optics can be used to improve the performance of electronic arithmetic units is the natural ability of optics to perform interconnections. There are several reasons for this and they have been documented in the excellent paper by Goodman et al.:¹³ (1) optical interconnects (OI) allow freedom from capacitive loading effects thus allowing high speed signal propagation, (2) OI offer immunity from signal interference effects which allows for massive 2-D interconnects, (3) OI do not have to be planar (as opposed to electronic interconnects), (4) if open space OI are used, then some reprogrammability can be achieved via "dynamic interconnections" and (5) optical signals can be injected directly into electronic logic devices.¹⁴

The above reasons clearly show the advantages of OI over their electronic counterparts. This chapter addresses the use of OI in processors that can be applied to the APAR problem. In Section 7.2 we briefly discuss the concept of array processors for matrix-matrix multiplication. In Section 7.3 we address some practical issues which are necessary for the realization of optically-interconnected array processors. Finally, in Section 7.4, we discuss the prototype optically-addressed ECL multiplier which we have fabricated.

7.2 Optically Interconnected Array Processor for Matrix Multiplication

A large number of algorithms (or parts of algorithms) used for the APAR problem can be expressed in terms of matrix multiplications. Typical examples can be found in some of the algorithms used for the eigensystem solution, presented in Section 2, as well as in the Gram-Schmidt technique^{1,2} presented in Section 9. Consider an example of matrix-matrix multiplication. Let matrices A and B each of dimensions $M \times M$ be represented by:

$$A = [a_{ij}] \quad (7.1)$$

and

$$B = [b_{ij}] \quad (7.2)$$

Their product C is given by

$$C = A \times B \quad (7.3)$$

where

$$c_{ij} = \sum a_{il} b_{lj} \quad (7.4)$$

Another way of expressing C is through outer products $A_i B_i$; i.e.,

$$C = [A_1 B_1 + A_2 B_2 + \dots + A_M B_M] \quad (7.5)$$

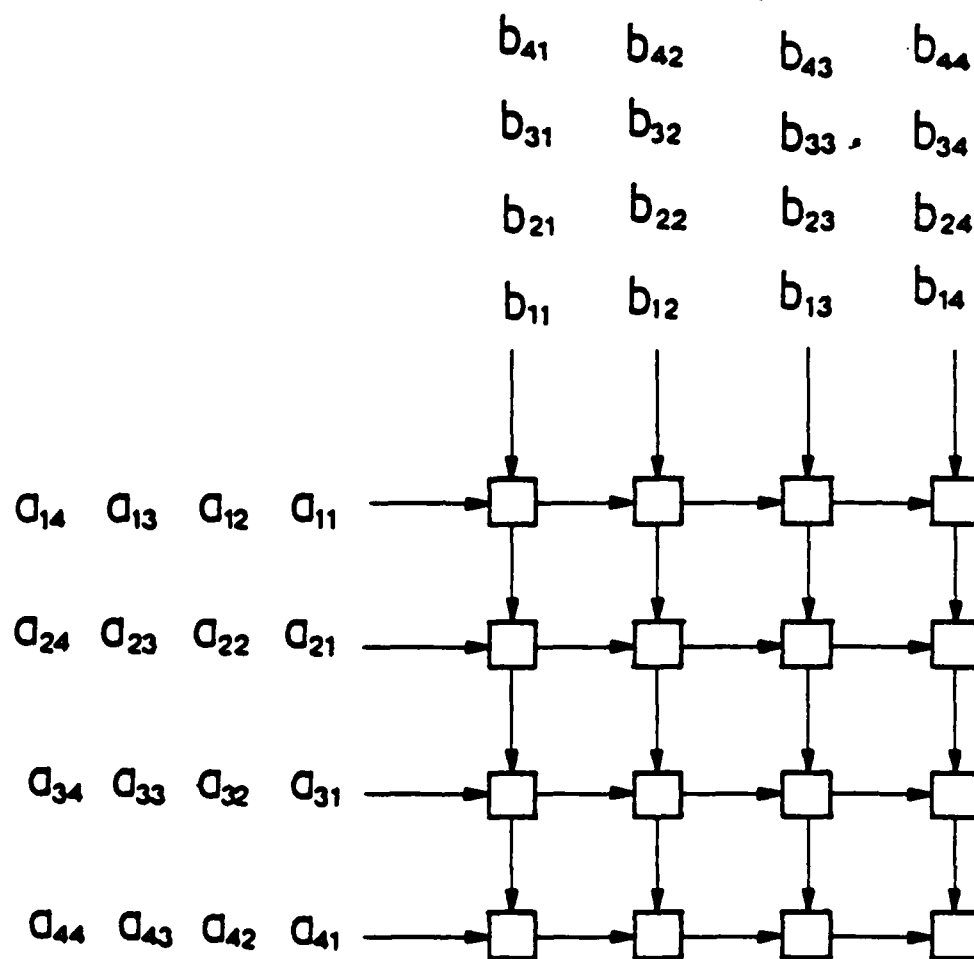


Figure 7.1 Square Array Processor for Matrix-Matrix Multiplication

A simple processor that implements Equation 7.5 is shown in Figure 7.1, and consists of a square array of $M \times M$ multiplier-accumulator units (MAU). In this system, the column data A_i and row data B_i are broadcasted instantly along the columns and rows of the array processor. At each clock cycle there is a new outer product formed which is consequently added to the previous product. In this fashion the total number of cycles needed for a full matrix-matrix multiplication is M . This design is not suitable for VLSI circuit design because it needs global communication.¹⁵ Thus, when VLSI implementation is desired, one has to configure the processor of Figure 7.1 in a systolic architecture so that only local interconnections are used. In such a scenario, the number of cycles needed for a full matrix-matrix multiplication depends on the specific systolic implementation used. For a square array, similar to the one of Figure 7.1, this number is of the order of $2M$. Thus the "globally interconnected" array of Figure 7.1 offers the advantage of improving the processing speed by a factor of 2. Note that when high frequency operation is desirable (≥ 500 MHz) the locally interconnected array processor probably requires the use of OI for data transmission. This is because at such frequencies the number, paths, lengths, and terminations of microstrip and strip lines are extremely critical in order to avoid effects such as capacitive loading, delays, overshoot/undershoot etc (an extensive but simple treatment of this issue can be found in Reference 16). Thus, since OI seem to be necessary for high frequency operation, it is logical to use OI in a global rather than local fashion.

7.3 Optical Interconnects for Array Processor

In this section we analyze the OI for the array processor of Figure 7.1. As we have described in the previous section, the column data A_i and row data B_i need to be broadcast instantly across the array. This implies that we need to address M different MAU inputs

with the same data. This in turn implies that we need to split the data 1:M. Thus, we need to address a simple technique which uses state-of-the-art technology and which allows us to drive M optical channels from one optical source. Before we describe such a technique it is of interest to discuss a technique for coupling the optical source (laser diode) into a fiber.

A schematic drawing of a fiber-optic adapter which we have developed for efficient coupling of the output power from a Mitsubishi ML4402 laser diode into a fiber of core $\geq 100 \mu\text{m}$ is shown in Figure 7-2. The output beam from the laser is elliptical with beam divergencies of 33° and 11° , full angular spread at the half-power points, along the major and minor axes, respectively. To collect this angular spread directly into a fiber without the use of lenses requires that the fiber end face be positioned much closer to the diode emitting surface than the windowed laser package allows. Accordingly, we have removed the window and protective can from the laser to allow complete access to the emitting surface.

For the fiber, we have chosen a glass fiber of 100μ core diameter and $140 \mu\text{m}$ cladding diameter, since this represents a fiber having one of the highest aspect ratios readily available. Although this factor is not critical for the multiplier application (see Section 7.4), it is an important parameter when the outputs from many fibers are combined in a fan-in or fan-out as is the case of the array processor or the case of the application to look-up tables discussed later in Section 8.3. Thus, to intercept all the laser output, out to the half-power points, into a $100 \mu\text{m}$ core fiber requires that the emitting surface-fiber face separation be less than $160 \mu\text{m}$. The fiber-optic adapter shown in Figure 7.2 accomplishes this, taking into account the manufacturing tolerance levels in the height of the diode surface above the base of the laser.

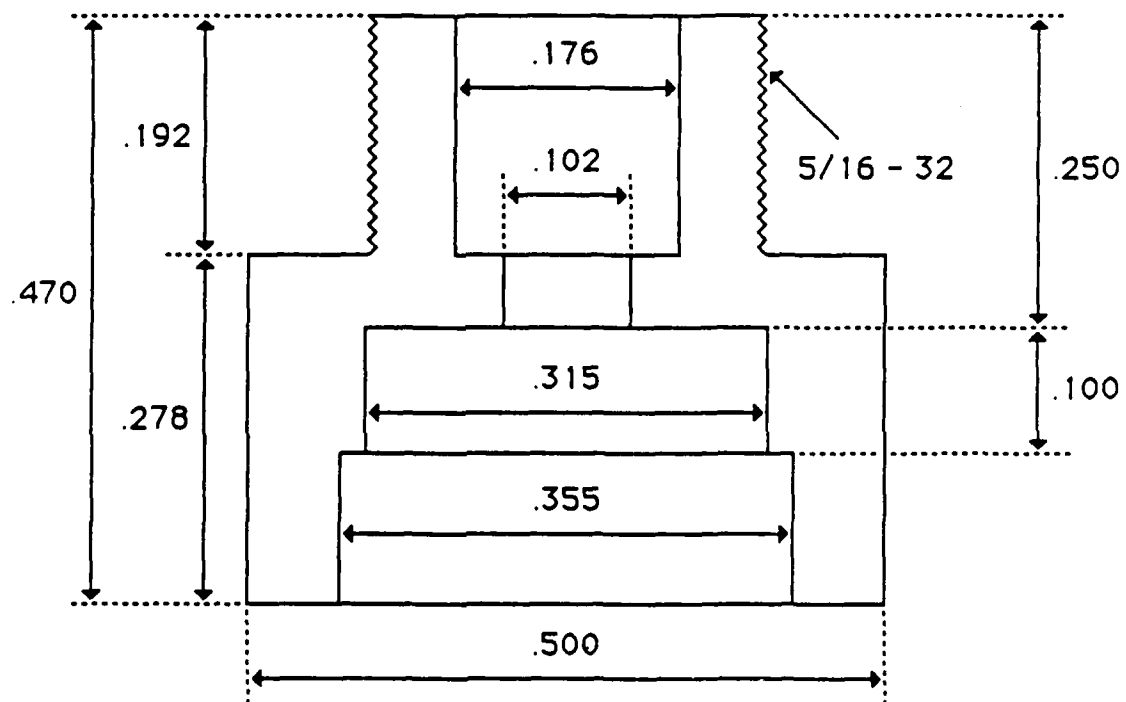


Figure 7.2 ML 4402 Laser Diode Pigtail Adapter

The adapter consists of a plexiglass rod bored at one end to be a slide fit to the diameter of the laser diode base. The diode is inserted into the adapter until it bottoms on the machined step in the adapter and is cemented to the adapter using UV-cured epoxy. In this way the diode emitting surface is precisely located (within diode manufacturing tolerances) with respect to the step in the plexiglass adapter. The cemented diode sits completely within the adapter to allow convenient attachment of the insulating adapter to a circuit board without danger of shorting the diode base (which forms one of the diode connections) to the board. The other end of the adapter is machined and threaded to accept (in this case) a standard Simplex ferrule connector. The length of the adapter above the step (which locates the base of the diode) is such that a ferrule, polished to its standard length, when inserted into the axial hole in the adapter is positioned on axis and with the face of the ferrule located within the distance of $160\text{ }\mu\text{m}$ required to intercept the divergent output beam of the laser. This adapter has proven to be a simple, efficient, and reproducible means of coupling the output from the laser diode into a $100\text{ }\mu\text{m}$ -core fiber. All of the hundred diode-adapter assemblies fabricated so far have given the full maximum diode output of 5 mW measured at the end of the fiber for diode currents $\leq 10\text{ mA}$ above the threshold current.

Splitting one optical channel into M optical channels can be achieved via the use of: (1) holograms, (2) star couplers in conjunction with fibers and (3) fiber-optic splitters that use resilient-ferrule connectors. The first approach is practical in applications where the M optical channels distribute their information in a relatively small area; e.g., $100\text{--}400\text{ cm}^2$. The second and third techniques allow signal distribution without any practical restriction in area or path length. Star-couplers, however, are more expensive and have a higher loss (for $M \approx 40$). For these reasons we have decided to use the resilient-ferrule connector approach (RFC).

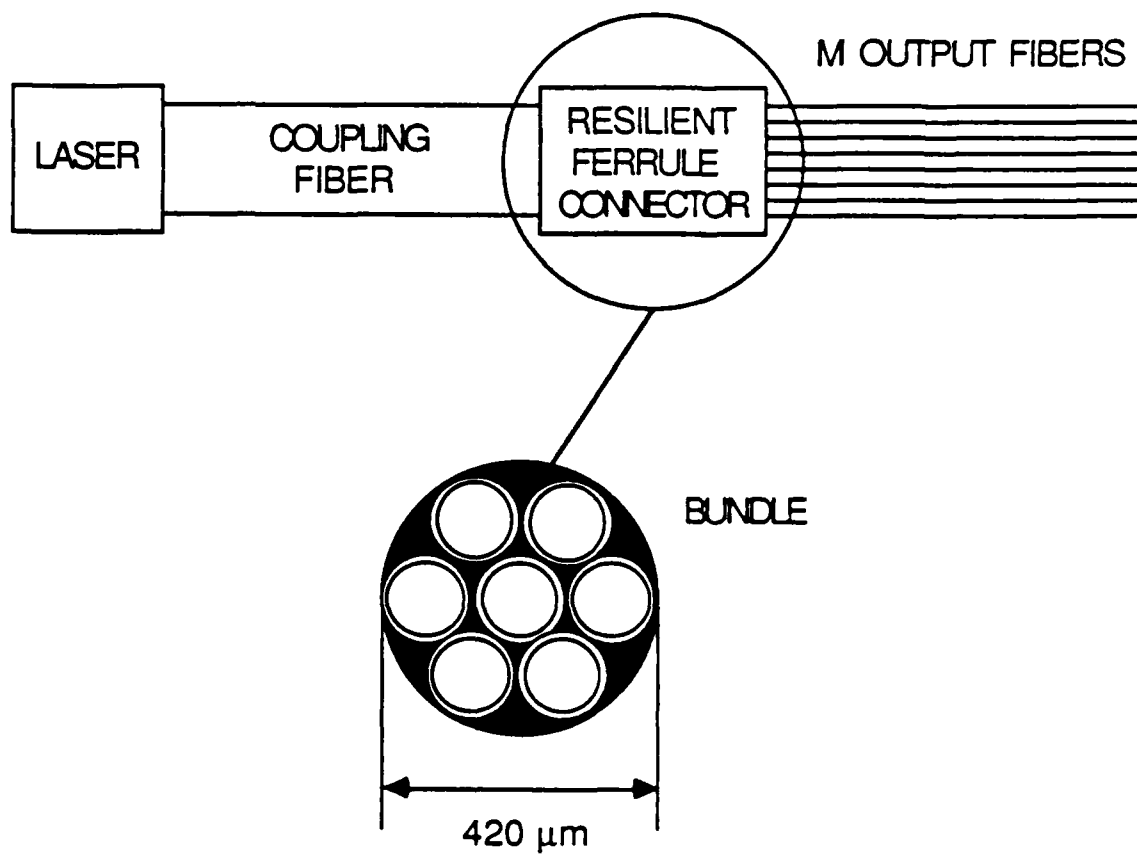


Figure 7.3 RFC Splitter Approach

In the RFC approach (Figure 7.3) the optical source, usually a laser diode, is coupled to a relatively wide core diameter fiber (e.g., 420 μm). The output of the wide fiber produces a uniformly distributed spot of light. We then pack a number of smaller diameter fibers into a resilient-ferrule connector (Figure 7.3), e.g., 7 fibers of 140 μm cladding and 100 μm core. Thus, we create an effective active area of 380 μm diameter which is covered by the cores of the fibers. Each fiber receives nearly the same amount of light. The other end of each fiber is terminated in a separate connector which is used for the MAU connection. Note that by reversing input/output ends, we can use the RFC arrangement as an M:1 combiner. This is exactly the way we use it for the look-up table residue approach we discuss in Section 8.3. Note that the RFC technique has losses that are comparable to those found in connectors (about 0.5 dB) and is less costly since it eliminates the expense of the coupler itself.

For efficient coupling, we need to maximize the core-to-cladding ratio (CCR) of the small fibers as well as the total effective receiving area of the fiber bundle in the RFC (shaded area in Figure 7.3). The former is needed in order to maximize the effective receiving area per fiber. The latter is necessary in order to minimize the amount of unused light. Maximizing the CCR implies that we avoid, if possible, the use of single-mode fibers which have a very small CCR of the order of 0.04 (e.g., 5 μm core and 125 μm cladding). Note, however, that single-mode fibers are the only choice if multi-Gb/s data rates are needed. In a multi-mode fiber, typical CCR's are of the order of 0.7 (e.g., 100 μm core and 140 μm cladding). For these fibers, and at $\lambda = 850 \text{ nm}$, the typical transmission bandwidth is about 100-200 MHz-Km. For our application, maximum distances of the order of a meter are expected. For these distances effective data rates of $\geq 1 \text{ Gb/s}$ can be easily achieved. In fact, we have shown that for 2 meters of 50112 AMP fiber, data rates of $> 1.2 \text{ Gb/s}$ can be achieved. Maximizing the total receiving area is equivalent to efficient fiber packaging. One can

NO-A182 874

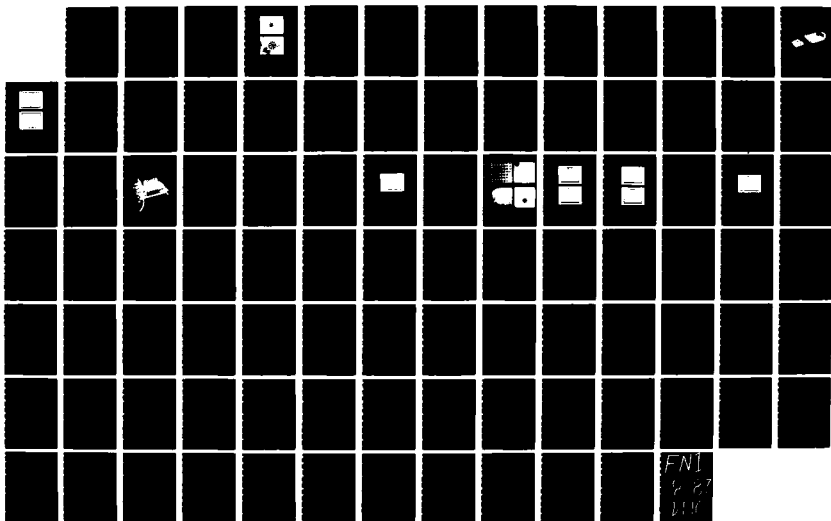
OPTICAL ARCHITECTURES FOR MATRIX PROCESSING(U)
 WESTINGHOUSE RESEARCH AND DEVELOPMENT CENTER PITTSBURGH
 PA A P GOUTZOULIS ET AL NOV 86 AFMIL-TR-86-1117
 F33615-84-C-1499

2/2

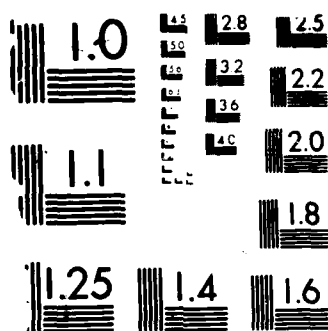
UNCLASSIFIED

F/G 12/6

NL



FNI
 887
 210



MICROCOPY RESOLUTION TEST CHART

U.S. GOVERNMENT PRINTING OFFICE: 1963 O - 348-094

easily show that for efficient packaging, a symmetric fiber arrangement similar to the one shown in Figure 7.3, is needed. In such an arrangement, the total number of fibers M_o and the overall diameter D are given by:

$$M_o = 1 + 3k (k + 1) \quad (7.6)$$

$$D = (2k + 1) d \quad (7.7)$$

where $k = 1, 2, 3, \dots$ and d is the cladding diameter. In this case the aspect ratio A_o of the total area is

$$A_o = [1 + 3k (k + 1)] a / (2k + 1)^2 \quad (7.8)$$

where $a = (d_1/d)^2$ is the aspect ratio of a single fiber of core diameter d_1 . In Table 7.1 we show, as a function of k , the total number of fibers M_o , the array diameter D , the overall efficiency η , and the core diameter d_2 , of the laser coupling fiber. For these calculations we assume that each fiber in the bundle has a 100 μm core and a 140 μm cladding, i.e., $a = 0.51$. For demonstration purposes, we have implemented the cases for $k = 1$ and 2 using 50112 AMP 100 μm /140 μm fiber. Typical output radiation patterns are shown in Figure 7.4. In both cases we obtain coupling efficiency results that are in excellent agreement with the figures of Table 7.1. Note that our experimental results show that for $M = 16$ and 19 there is little difference in η . Thus, for all practical calculations involving $M = 16$ one can use the $M = 19$ data. Also note that because of practical reasons (availability of proper fibers, connector dimensions, etc.) $M = 19$ is probably the upper limit of the RFC technique.

Table 7.1

k	M_o	D (μm)	η	d_L (μm)
1	7	420	0.40	380
2	19	700	0.39	660
3	37	980	0.39	940
4	61	1260	0.38	1220

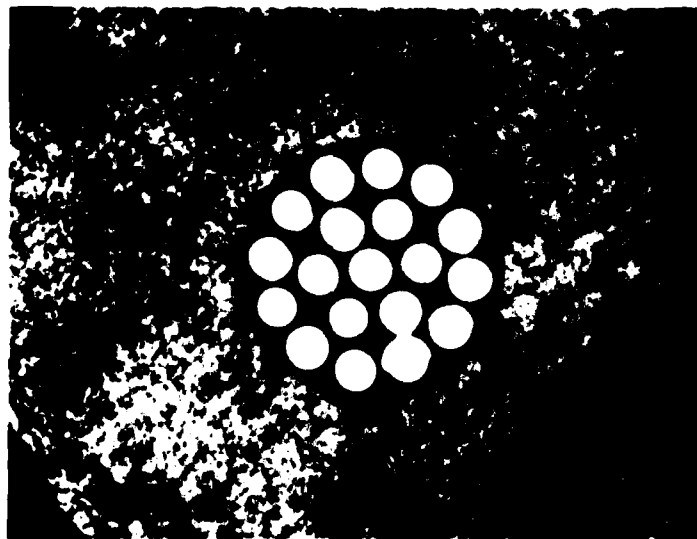
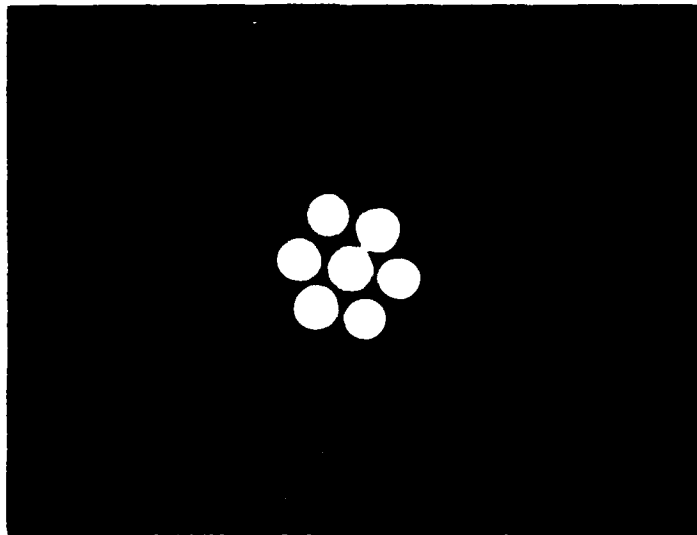


Figure 7.4 Output Radiation Pattern from 1:M RFC Splitters with $M = 7$ and 19.

We now discuss some issues associated with the amount of optical power necessary for the interconnections. Let us assume that we are dealing with a 16×16 array processor which we intend to interconnect using discrete detectors. For high data rates (e.g., several hundreds of MHz) we need the detector to interface to an impedance of about 50 ohms. Thus, for a 1V swing, the detector needs to provide 20 mA. Current state-of-the-art high speed (rise time ≤ 0.5 nsec) pin diode detectors (e.g., Motorola MFOD 1100) have a responsivity of $0.3 \mu\text{A}/\mu\text{W}$. Thus, we require about 67 mW of optical power incident on the detector. Since $M = 16$, the individual fiber coupling efficiency is $0.39/16 = 0.024$. Thus, the total optical power incident on the RFC needs to be $67/0.024 \text{ mW} = 2.8 \text{ W}$. This power is beyond that obtainable from state-of-the-art laser diodes so that buffers must be incorporated between detectors and MAU inputs. A typical example of such a buffer unit is the Advance Micro Devices Am 6687 comparator which allows for data rates of up to 300 MHz. This device requires a minimum input of 5 mV which corresponds to a total laser power of 14 mW. State-of-the-art low-cost laser diodes, such as the Hitachi HLP 1400, can deliver up to 20 mW at data rates in excess of 800 Mb/s. It is thus our conclusion that if the discrete detector/buffer approach is used in conjunction with existing, low cost technology, then the OI for a fully parallel 16×16 array processor with data rates in excess of 300 Mb/s per channel can be built. If the electronic MAU's are capable of following the above data rates, then the system MS will be 300 MHz and the total throughput rate will exceed $16 \times 16 \times 300 \times 10^6 \text{ M-A/s} = 76 \times 10^9 \text{ M-A/s}$ which is obviously a tremendous processing capability. Note that if the detectors can be integrated with the MAU chip, then the effective impedances will increase perhaps by an order of magnitude. In this case the buffers are not needed and the OI that consists of existing, low-cost laser diodes/RFC/detectors can deliver data rates of the order of 1 Gb/s.

In the following section we present an experimental, optically addressed, 4 x 4 bit ECL MAU that uses components similar to the ones we have been considering.

7.4 Prototype Optically Addressed ECL Multiplier

To illustrate the capability of optical interconnections we have fabricated a 4x4 bit optically addressed multiplier based on ECL logic. The complete arrangement is shown schematically in Figure 7.5.

The optical data generator consists of eight pulsed laser diodes (Mitsubishi Type ML 4402) which provide the two 4-bit words. These laser diodes are driven from a common pulse-generator source (Hewlett Packard Type 8082A) which is fanned out to eight lines each of which is connected to the transistor drive (Motorola Type 2N5943) of each laser diode. Each laser diode is housed in a fiber optic adapter (described in Section 7.3) which accepts a standard Simplex fiber connector. This optical data generator may be driven at the maximum frequency of the pulse generator (250 MHz for a 50% duty cycle) thereby providing the equivalent of a 500 MHz binary (0,1) data rate.

The optical interconnect consists of eight fiber optic lines fabricated from 100 μm core/140 μm cladding cable (AMP 50112). These lines are provided with Simplex connectors at either end for coupling the laser diode output from the data generator board to the optical interface/ECL multiplier board.

A schematic diagram of the circuit used for the optical interface/multiplier is shown in Figure 7.6. The optical interface is provided by pin diode-comparator combinations. Each optical input signal corresponding to a single binary bit is fed to a pin diode detector (Motorola Type MF0D 1100) housed in a Simplex fiber connector mount to accept the fiber-optic interconnection. The output from each pin diode

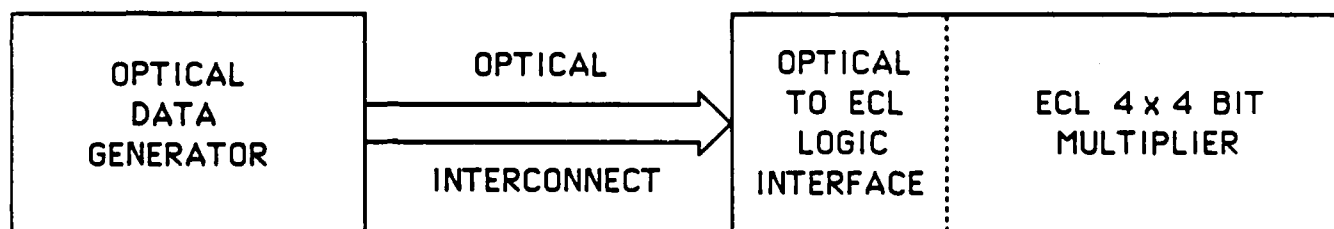


Figure 7.5 Schematic Diagram of the Optically Addressed Multiplier

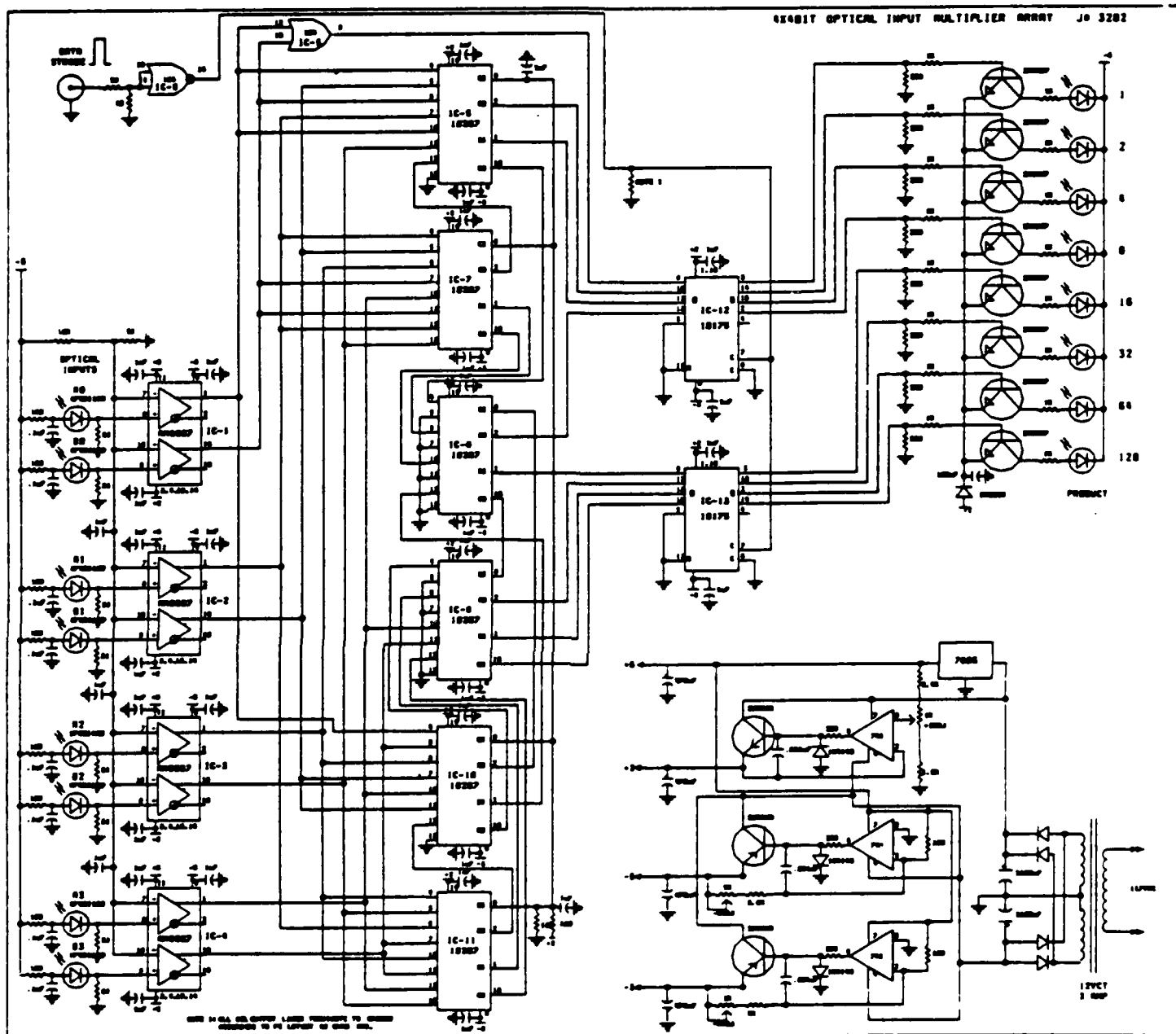


Figure 7.6 4x4 Bit ECL Multiplier Design

is in turn fed to a comparator (AMD Type AM 6687) which generates a standard ECL logic pulse provided the amplitude of the output pulse from the pin diode exceeds a preset threshold level.

Insofar as the multiplier itself is concerned, there are several architectures available for configuring electronic gates to perform multiplication.¹⁷ For the present demonstration we have chosen to use the Pezaris arrangement¹⁸ which utilizes full adders as shown schematically in Figure 7.7. Thus, for two 4-bit binary numbers $A = a_3a_2a_1a_0$ and $B = b_3b_2b_1b_0$, the product $Z = A \cdot B = (a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0)$ may be written in the form

				a_3b_0	a_2b_0	a_1b_0	a_0b_0
			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
		a_3b_1	a_2b_1	a_1b_1	a_0b_1		
	a_3b_2	a_2b_2	a_1b_2	a_0b_2			
<hr/>							
Z_7	Z_6	Z_5	Z_4	Z_3	Z_2	Z_1	Z_0

corresponding to the practical implementation of Figure 7.7.

The building block of the multiplier is a 2x1 bit array multiplier from the MECL Series (Motorola Type MC 10287) which is a dual package each half incorporating two input AND gates, for forming the binary bit products followed by a full adder for summing the products, with internal carry lookahead for high speed operation. The logic diagram of the array multiplier block is shown in Figure 7.8 and is particularly suited for use in the Pezaris architecture. Thus, the data output from the interface comparators are fed to the inputs of six MC 10287 packages the outputs of which provide the products Z_1 to Z_7 . The least significant bit product $Z_0 = a_0b_0$ is provided by a single AND gate.

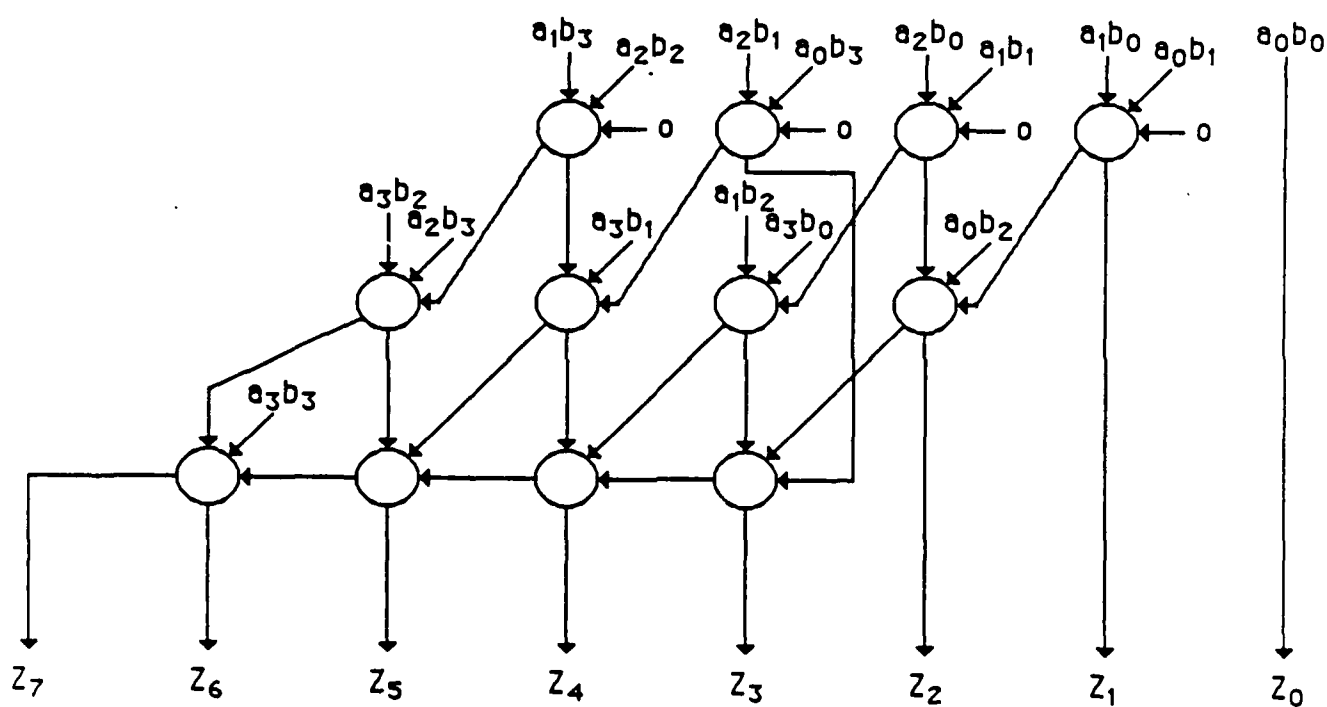


Figure 7.7 Schematic Diagram of the Pezaris Multiplier Arrangement

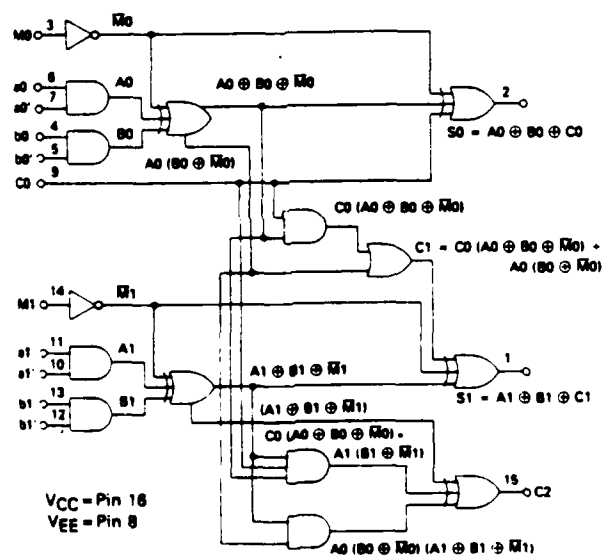


Figure 7.8 Logic Diagram of the MC 10287 Array Multiplier Block

A strobed latch/LED display provides visual readout of the product from the multiplier. Each of the product bits from the multiplier is fed to a latch (Motorola Type 10175) which is strobed by a suitably delayed pulse derived from the same pulse generator used to produce the optical data. Thus, the latches are strobed synchronously with the data, each output pulse from the latches driving the specific LED associated with each product output bit.

Layout and fabrication of the double-sided boards was carried out on a CAD/CAM facility. In the board layout, particular attention was paid to the design to ensure high-speed operation. A photograph of the assembled boards is shown in Figure 7.9.

For the purpose of exercising the multiplier, the input words A and B are varied by connecting the appropriate fiber-optic lines between the data-generator and multiplier boards. A composite record showing the input/output pulse responses from the multiplier board is shown in Figure 7.10(a). The upper trace on this record corresponds to the input data-bit pulse from the interface comparator (all eight pulses are coincident in time) followed, in sequence, by the eight output bit pulses corresponding to the products Z_0 to Z_7 , respectively. This record has been obtained by varying the input word A keeping the full input word B (i.e., $b_3 = b_2 = b_1 = b_0 = 1$) fixed. From Figure 7.10(a) it may be seen that the delay time between the input data pulse and the output product pulse increases from Z_0 through Z_6 with the most significant bit Z_7 (the final carry bit) delayed somewhat less than Z_6 . The various delays measured from Figure 7.10(a) are compared with those expected from the multiplier architecture used in Table 7.2. For the latter, we have characterized the delays in terms of the unit gate delay time Δ . Thus, the propagation delay times of the AND, OR, and XOR gates incorporated in the multiplier blocks are 2Δ , 2Δ and 3Δ , respectively. Taking a value of $\Delta = 0.3$ ns (corresponding to a propagation delay of an ECL NAND gate), we obtain the values shown in the last column of

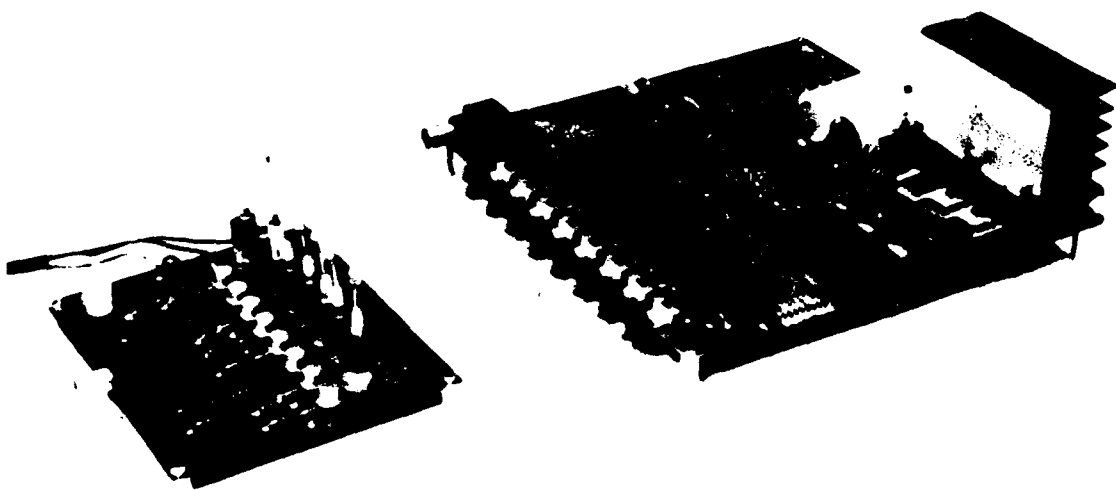


Figure 7.9 Photograph of the Assembled 4x4 Bit Multiplier Board with Optical Interconnects from the Data Generator Board

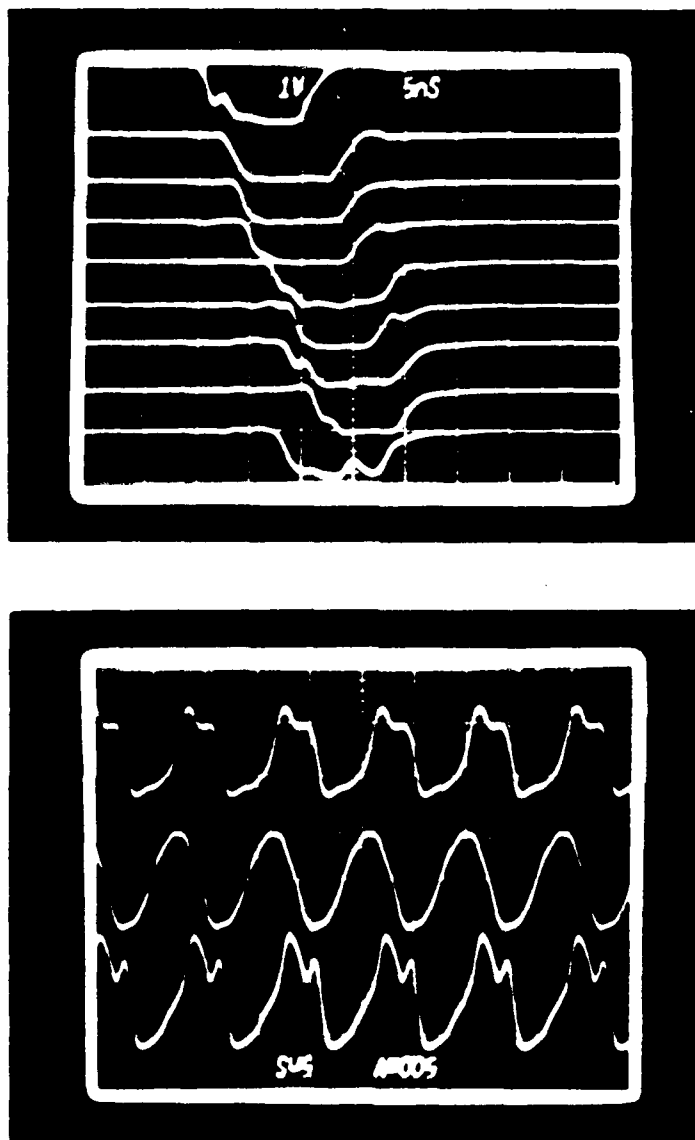


Figure 7.10 (a) Composite oscilloscope record showing in sequence from top to bottom, the relative delays between the input data bit pulse, and the various output bit pulses corresponding to the products Z_0 to Z_7 , respectively.

(b) Composite oscilloscope record showing in sequence from top to bottom, the input data pulse and output bit pulses corresponding to Z_0 (minimum delay) and Z_6 (maximum delay) for maximum operating speed of the multiplier (220 MHz).

Table 7.2

Input to Z_n Propagation Delay Times

	Measured (ns)	Estimated	
		Δ	ns ($\Delta = 0.3$ ns)
Z_0	2	2	0.6
Z_1	3.5	8	2.4
Z_2	4.5	16	4.8
Z_3	6.5	26	7.8
Z_4	8.5	30	9.0
Z_5	8.5	31	9.3
Z_6	11	35	10.5
Z_7	7.5	33	9.9

Table 7.2. These are in good agreement with the measured values. The total multiplication time is dependent on the value of the input words. The maximum time is evidently that for which the product includes the 2^6 (Z_6) bit. For the present arrangement this propagation delay is measured to be 11 ns which is in reasonable agreement with the manufacturer's specification of 14 ns.

In characterizing the performance of multipliers we wish to distinguish between the total multiplication time and the throughput rate. In the present arrangement, the maximum throughput rate is governed by the requirement that the data bit pulse be sufficiently wide that all of the product bit pulses be present during the strobe pulse. Thus, the minimum pulse width is given by the maximum difference in propagation delay among the product bit pulses, i.e., the difference in propagation delay for Z_0 and Z_6 . From Table 7.2 this is measured to be 9 ns. Thus, the present multiplier has a maximum data throughput rate of 110 MHz. However, we note that if a truly pipelined architecture is used the throughput rate may be significantly increased. While we are not able to do anything in the present demonstration at the chip level we note that we can exercise the board at a higher data throughput rate than is dictated by the strobe requirements. An example of this is shown in Figure 7.10(b) which shows from top to bottom, the input data pulse, the Z_0 output bit pulse, and the Z_6 output bit pulse, respectively, where the input data pulse width has been minimized while still maintaining all output bit pulses. From Figure 7.10(b) it may be seen that a data throughput rate of 220 MHz is possible if suitable delays are introduced in the circuit so that the output bits can be strobed simultaneously. This throughput rate is limited both by the comparator delay time in the optical interface and by propagation delays in the multiplier chips themselves.

In conclusion, we see that existing fiber-optic and electronic technology can be used in order to fabricate a fully parallel optically

interconnected square array processor for matrix-matrix multiplication. With this existing technology we expect multiplication speeds that exceed 200 MHz (assuming a pipelined architecture). Such an array processor is obviously easier to implement than the Acousto-Optic processors we have presented in Section 4. This further demonstrates that Acousto-Optic systems cannot compete with existing electronic technology.

8. RESIDUE LOOK-UP TABLE ELECTRO-OPTIC PROCESSING

8.1 Introduction

In this section we discuss a residue arithmetic approach for high-speed Electro-Optic processing. As we have shown in Section 5, Acousto-Optic binary processors cannot compete efficiently with existing digital electronic counterparts. One of the reasons is that Acousto-Optics performs only part of the operation, i.e., the convolution, and power-hungry electronics is needed in order to convert the mixed-binary data of the convolution into a conventional binary form. If operation in the mixed-binary form would be possible for many processing steps, then the system efficiency would improve because of the fewer conversions that would be necessary. Unfortunately, this is not the case. Thus we have decided to explore other arithmetic schemes which allow many operations to be performed before conversion into a more conventional arithmetic is needed. One such possibility is residue arithmetic.

In the following Section 8.2 we present a brief discussion of the basics of residue arithmetic. In Section 8.3 we describe a possible look-up table (LUT) technique for high speed processing and in Section 8.4 we discuss our prototype LUT. In Sections 8.5 and 8.6, we show how one can convert from binary-to-residue and from residue-to-binary, via utilization of LUT techniques. In Section 8.7, we discuss issues associated with hardware minimization. Finally, in Section 8.8 we discuss an example of residue LUT processing, a square array processor for matrix-matrix multiplication.

8.2 Residue Arithmetic Basics

Residue arithmetic, because of lack of carries, is probably the fastest way of performing addition, subtraction, multiplication and various polynomial transformations.¹⁹ The residue number system (RNS) is based upon N fixed relatively prime integers m_1, m_2, \dots, m_N which are called moduli (or base). An integer number X , that lies in the range 0 to $(M-1)$ (M is the product of the N moduli) is uniquely represented with respect to the N moduli via the N -tuple of residues $(R_{m1}, R_{m2}, \dots, R_{mN})$. Each residue R_{mi} is defined to be the least positive integer remainder by the division of X by m_i . For example, for the 5 moduli 7, 9, 11, 13, 16 the maximum range M is equal to

$$M = 7 \times 9 \times 11 \times 13 \times 16 = 144,144 \quad (8.1)$$

Thus, this set of moduli allows us to represent any integer in the range 0-144,143. For example, 279 is represented by (8,0,4,6,7). Note that it is convenient to have an even modulo so that we can detect negative numbers easily. In this case, the range 0- $(M/2 - 1)$ is used to represent positive numbers, whereas the range $M-1$ to $M/2$ is used to represent negative numbers. Note that in the latter case M must be subtracted in order to obtain the correct answer.

To perform residue arithmetic operations,¹⁹ we first convert all the numbers of interest into RNS. We then perform the arithmetic operation by operating on their RNS representations. The specifics of the RNS operation depends on the specific arithmetic operation we are performing. In all cases, however, operations over different moduli are independent, there are no carries and the result of the operation on modulo m_i cannot exceed m_i-1 .

To add in RNS we simply add the corresponding residues and then we find the residues with respect to each modulo. For example, in base (7,9,11,13,16) the sum $279 + 31 = 310$ (31 in RNS is (3,4,9,5,15)) is

$$\begin{array}{r} (6,0,4,6,7) \\ + \\ (3,4,9,5,15) \\ \hline \end{array}$$

$$(2,4,2,11,6) = 310.$$

To subtract in RNS we change each residue digit of the subtrahend by its complement and then perform an addition. In RNS the complement of a residue is its difference from the modulo; e.g., in base 13 the complement of residue 9 is 4. For example in base (7,9,11,13,16) the difference $279 - 31 = 248$ in RNS is

$$\begin{array}{r} (6,0,4,6,7) \\ + \\ (4,5,2,8,1) \\ \hline \end{array}$$

$$(3,5,6,1,8) = 248$$

To multiply in RNS we multiply the residues at each modulo and then find the resulting residue. For example, $279 \times 31 = 8,649$ is

(8,0,4,8,7)

x

(3,4,9,5,15)

(8,0,3,4,9) = 8,649

Division in RNS is not always possible. RNS by definition represents integers. The division of two integers is not always an integer and thus it cannot be represented in RNS. However, in some special cases, division can be performed by means of multiplicative inverses. An integer Y is called the multiplicative inverse of X if the product YX with respect to modulo m is 1. For example, in modulo 11 the multiplicative inverse of 5 is 9. To use multiplicative inverses for division, the result of the division must be an integer and the divisor must not contain any moduli as factors.

8.3 Residue Look-Up Table Processing

Because of the lack of carries, residue arithmetic allows independent calculations per modulo without the need for different modulo processors to cross-communicate. A general RNS processing scenario is shown in Figure 8.1. In such a general scenario, binary data are fed into a binary-to-residue converter (B/R). The converter feeds its outputs (N outputs for N-moduli operation) to N different processors. All processors perform exactly the same RNS function(s) but with respect to a different modulo m_i . Upon completion of the operation, the outputs from the N processors are fed into a residue-to-binary converter (R/B), whose output is the result expressed in a conventional binary form.

Dwg. 9384A77

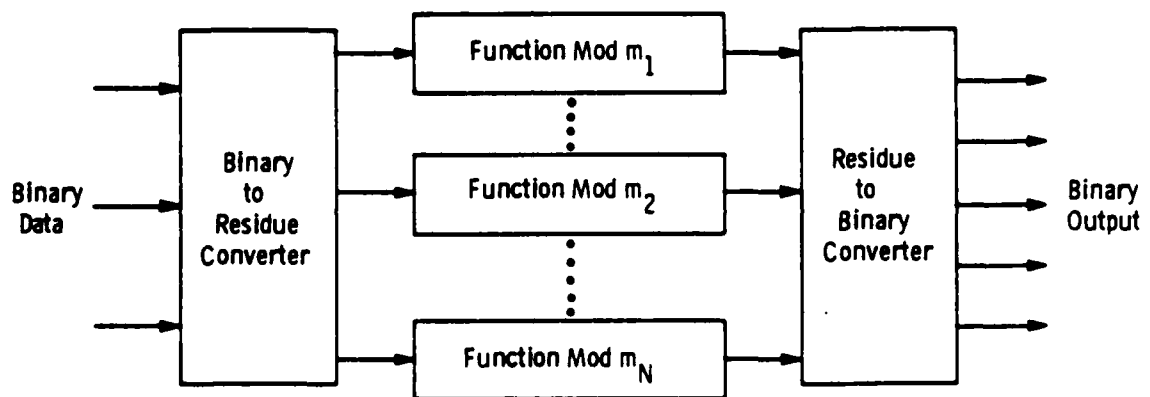


Figure 8.1 General RNS Processor Configuration

Because of this inherent lack of carry propagation, high speed processing can be achieved. In addition, all processors are similar and no global interconnections are needed (a very important consideration in a VLSI configuration). Since the system dynamic range is the moduli product, high dynamic range is achievable by using more parallel channels. Thus, high dynamic range can be achieved without reduction of the processing speed.

Let us now concentrate on the RNS processing itself (B/R and R/B conversion is discussed in detail in Section 8.5 and 8.6) under the assumption that residue representations are available. Because of the lack of carries, the bounded output range, etc., in RNS, various arithmetic operations may be implemented via the use of the look-up table (LUT). The idea is illustrated in Figure 8.2 where the LUTs for multiplication and addition in modulo 5 are shown. For the multiplication LUT, the objective is to create the product of numbers X and Y without performing an actual multiplication. For modulo m_i , the LUT has two sets of inputs (one for X and one for Y), each of which consists of m_i inputs (X or Y can take only m_i different values). For different values of the inputs, we obtain a different value at the output. These outputs are pre-calculated and stored and upon interrogation of the LUT with the inputs are read out.

Various forms of LUT implementation have been proposed and analyzed in the literature in the last decade. Among the architectures suggested are those of Huang et al,²⁰ Tai et al²¹ and Polky et al,²² which are more or less designed around electro-optic control of beams propagating in integrated optical waveguide structures. Another approach is the one by Gaylord et al,²³ which is based on binary coded residue LUTs. Such a processor exploits the multiple parallel channel processing capability that is inherent in optical systems. It performs EXCLUSIVE OR and NAND logic operations through the use of optical LUTs, which are based on holographic recordings or the use of spatial light modulators.

-

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

A) MULTIPLICATION

-

	0	1	2	3	4
0	0	1	3	2	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

B) ADDITION

Figure 8.2 Look-up Tables for Multiplication and Addition in Modulo 5 Residue Arithmetic

Our approach for implementing residue LUTs is based on the utilization of small, high-speed light emitting diodes (LEDs) or laser diodes (LDs) in conjunction with fiber-optic combiners or holograms. Numerical operations are performed simply by generating a light pulse which reaches a detector that has been encoded for the number resulting from each operation. Thus, for the modulo 5 multiplication (Figure 8.2) light produced at the intersection of inputs 3 and 2 drives the detector labelled 1. Similarly, for the modulo 5 addition, the light generated at the intersection of 3 and 2 illuminates a detector encoded 0.

One way for implementing this concept is through an interlaced two-dimensional grid of electrodes in conjunction with high-speed LEDs or LDs at the intersection points (Figure 8.3). A voltage pulse applied to each input line, such that the pulse amplitude is less than the LED junction voltage but that twice the pulse amplitude exceeds it by a considerable margin, causes the diode at the intersection point to emit strongly. The emitted light is transmitted to a detector that is encoded for the number to be produced at that table location, as indicated by the number in each grid box. To minimize the number of detectors required and to promote flexibility in LUT geometry, we use fibers (or a hologram) to transmit light from each diode that corresponds to a given digit to the single detector encoded for that digit.

Other arithmetical operations use the same LUTs in combination with one-sided subprocessors (or wiring maps) which precondition some of the inputs to these LUTs. Thus, subtraction proceeds through formation of the additive inverse of the subtrahend followed by look-up of the difference in the addition table, while division proceeds via the multiplication table after first forming the multiplicative inverse of the divisor. The wiring maps which form these inverses in modulo 5 are shown in Figure 8.4. Additional operations such as raising to an

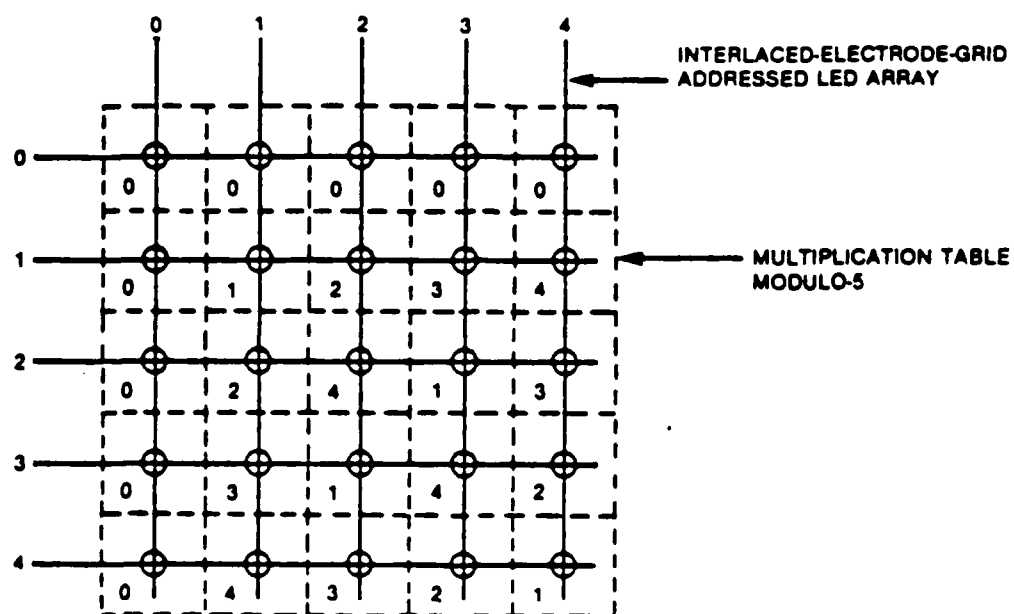


Figure 8.3 Schematic Representation of a Modulo 7 Multiplication LUT with LED Light Sources as Look-up Indicators. The numbers in each block represent the detector assigned to that block.

0	1	2	3	4
1	1	1	1	1
0	4	3	2	1

A) ADDITIVE INVERSE, \bar{a}

0	1	2	3	4
	1	1	1	1
	1	3	2	4

B) MULTIPLICATIVE INVERSE, a^{-1}

Figure 8.4 Look-up Tables (Wiring Maps) for Additive and Multiplicative Inverses in Modulo 5 Arithmetic

integral power or an inverse integral power are also performed using wiring maps which do not require generation and detection of light pulses. This shows the increased flexibility of the RNS LUTs as compared with DMAC and BPAM. The inputs and outputs of the LUT remain in the same residue base and thus different LUTs can be easily interconnected without the need for conversion. Such capabilities allow fast data flow and ease of pipelining as we show in Section 9.

We now estimate the expected performance of the LUTs. MS figures of the order of 1 GHz are to be expected since laser diodes have already achieved sub-nsec switching times²⁴ and operation of LEDs at frequencies > 1 GHz has also been reported.²⁵ To estimate the power consumption we must consider a specific computation example such as the multiplication of two 8-bit numbers. The possible product range (6.5×10^4) can be covered with the moduli 3, 5, 7, 8, 11 and 13. Assuming 1 GHz operation, a signal/noise ratio of ~ 30 (corresponding to a negligible bit error probability) and the use of detectors with NEP of $\sim 10^{-13} \text{ W}/\sqrt{\text{Hz}}$, we find that the optical power incident on the detector is about $0.1 \mu\text{W}$. Furthermore, assuming ~ 10 dB fiber coupling losses and 1% diode conversion efficiency we find that the total electrical power per operating diode is $\sim 100 \mu\text{W}$. If 10% of this power is used for diode prebiasing then the total prebias power consumption is about 4.4 mW. Adding to this figure the power required to turn on the proper diodes ($2 \times 47 \times 100 \mu\text{W} = 10 \text{ mW}$) we find that the total power consumption of the diodes is $\sim 14.4 \text{ mW}$. Next, we calculate the power consumption necessary to drive a LUT from another LUT, which is usually accomplished via the use of buffers. This is an important issue which needs to be investigated in detail but preliminary results suggest that only one buffer unit per LUT (the one connected to the "on" detector) need be on at any time. Assuming this to be the case and that the buffer unit is only 1% efficient, we find that the total buffer power consumption is $6 \times 100 \times 100 \mu\text{W} = 60 \text{ mW}$. Thus, the total

power consumption is 75 mW which when associated with a MS of 1 GHz, corresponds to a SE of $\sim 1.3 \times 10^{10}$ M-A/sec. W.

This preliminary analysis suggests that the LUT structures considered here may offer MS and SE advantages of about an order of magnitude over those of DMAC, BPAM and GaAs processing units.

8.4 LUT Experimental Results

To demonstrate as well as further understand the LUT concept we have fabricated a modulo 7 LUT which can be configured either as an adder or as a multiplier by changing the fiber-optic connections (Figure 8.5). The laser diodes used, Mitsubishi ML 4402, are capable of providing 5 mW pulses of light output at 780 nm. Their threshold current is between 35 mA and 40 mA and the operating current is about 50 mA. The LUT format is a square matrix of 7x7 LDs arranged in 7 rows and 7 columns as shown in Figure 8.6. A double-sided FR4 board is used which allows the implementation of a "non-additive" scheme in which rows are connected with common anodes and columns are connected with common cathodes. In an alternative scheme the cathodes (anodes) are grounded and the anodes (cathodes) are connected to both row and column lines. Note that in this "additive" scheme we need to decouple row and column lines (via the use of diodes) in order to avoid spread of the drive current pulse. For this reason and because of the more complex interconnection patterns of this scheme, we have decided to use the "non-additive" scheme.

To ensure high-speed operation we have decided to use ECL compatible drivers. Current to each anode row is supplied through the 35 Ohm resistors R_1 - R_7 which are connected to + 5 V. This row current is diverted from the laser diodes by transistors Q_1 - Q_7 . When the row inputs to these transistors are at ECL logic "1" ($\sim + 1.2$ V) the

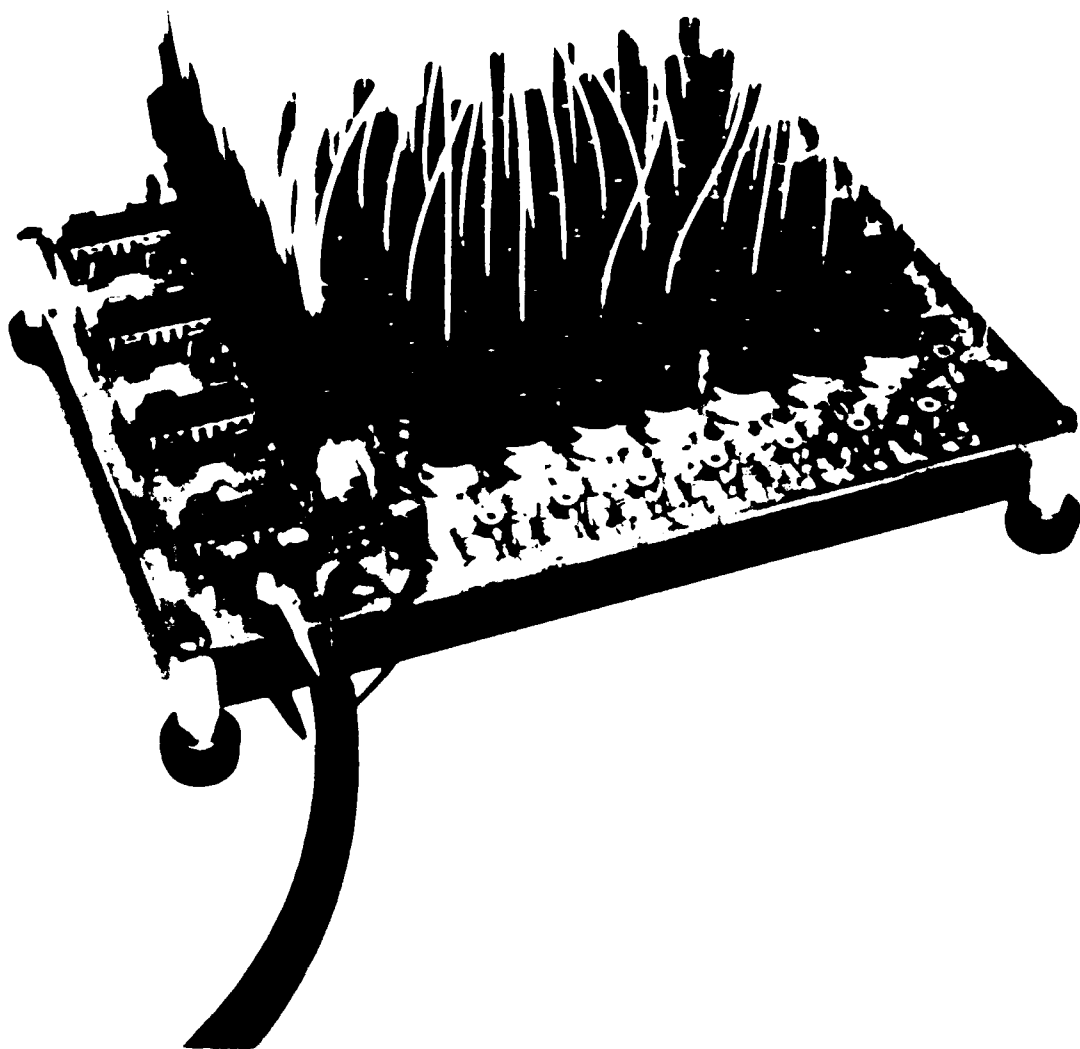
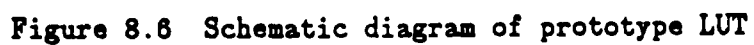


Figure 8.5 Photograph of prototype modulo 7 LUT

DETECTORS



transistor diverts about 55 mA from each row. Each cathode column is connected to ground through a transistor, Q_8-Q_{14} , which can sink about 70 mA when an ECL logic "1" is present at the input, and through a parallel 510 Ohm resistor which provides the path for initial bias current to the diodes. When all lasers are "off", i.e., rows at logic "1" and columns at logic "0", current through row resistors R_1-R_7 is about 67 mA. The remaining row current (≈ 12 mA) not diverted by Q_1-Q_7 flows through that row of lasers via the column resistors. This initial current rises to ≈ 25 mA when a row transistor is turned off. It should be noted that the sum of all row currents not diverted by input transistors Q_1-Q_7 and not flowing in the column resistors must be sunk by column transistors, Q_8-Q_{14} , when that column is addressed.

The output from the laser diodes is coupled into RFC fiber-optic 7:1 combiners (see Figure 8.5) described previously in Section 7.3. The combiner output drives a high speed (rise/fall times ≤ 1 ns) pin diode (Motorola MOFD 1100) which in turn drives a high speed comparator (AM6687). The comparator's dual outputs (positive and negative) are capable of driving any of the Q_1-Q_{14} transistors. In this scenario, we simulate a LUT that is capable of driving another LUT.

An important factor for high speed operation is the operating mode of the driver transistors Q_1-Q_{14} which need to provide both high switching speed and adequate current levels. Initially, Motorola 2N5943 transistors were used in a switching mode which offers the advantage of large current flow. In this operating mode, a drive pulse applied to the base quickly saturates the transistor and thus adequate current flow is achieved. The problem associated with this approach is the slow turn-off times due to the relatively long storage times. Use of Schottky diodes between base and collector reduces the turn-off times but not enough to allow GHz-type operation. With this technique we are able to demonstrate about 110 MHz rates. An alternative approach is to use the transistors in a current mode which virtually eliminates

the problem of slow turn-off times. Note, however, that this technique allows relatively small amounts of current flow and is, thus, suited to relatively low-current situations. Nevertheless, we have implemented this technique using Motorola MRF 581 transistors which have an F_t of 4 GHz at 100 mA collector current levels. These transistors allow operating current levels of 45-50 mA and about 2 mA of pre-bias current which is shared among 7 laser diodes.

Our first experiment deals with the effect of the board's micro-strip lines on switching speed. For this purpose only two laser diodes are connected at positions (1,1) and (7,7). These positions are subject to the smallest and longest propagation delays, respectively. The top trace of Figure 8.7 shows the 250 MHz RZ ECL waveform used to drive the laser diodes. This is derived from a Hewlett-Packard (Type 8082A) pulse generator. The second trace of Figure 8.7 shows the (7,7) laser diode response to that waveform (detected through an MFOD 1100 pin diode) when the laser cathode is pulsed and the anode is DC biased. Similarly, the third trace shows the response when the anode is pulsed and the cathode is biased. Finally, the fourth trace shows the response when both anode and cathode are pulsed simultaneously. As these data show, the laser diode responds to at least 250 MHz RZ or 500 MHz NRZ data rates. (These frequency limits are dictated by the available pulse generator). Note the reduction in pulse width, by about a factor of 2, when the anode is partially or fully driven. This behavior is not well understood and is believed to be associated with the specific laser diodes used. In any event, this effect is not believed to seriously affect the LUT's performance. Note that the (1,1) laser diode shows behavior similar to the (7,7) laser diode with the exception of the lack of a small delay (≤ 1 ns). These results show that both the impedance and inductance of the board's micro-strip lines allow for at least moderately high switching speeds.

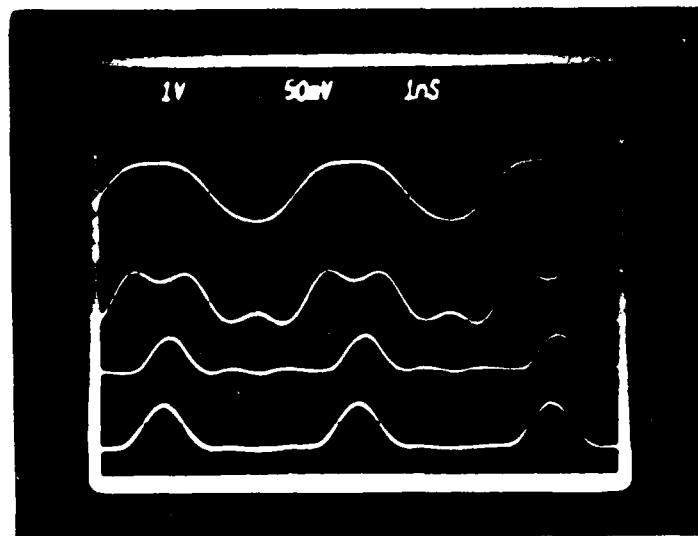
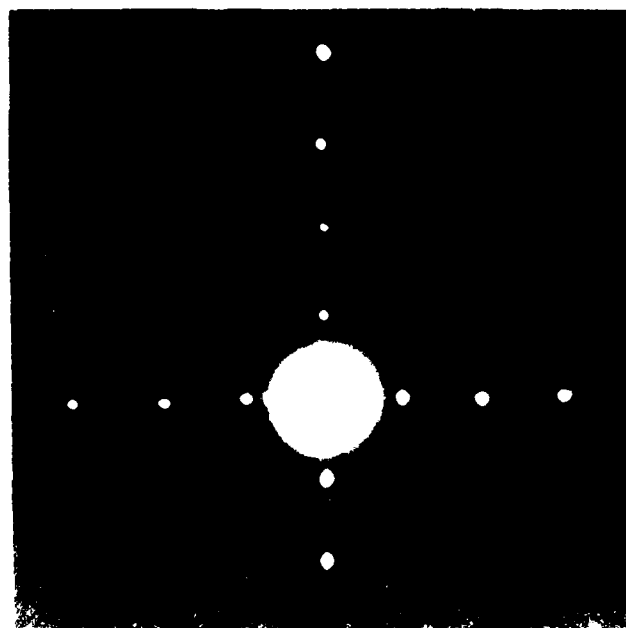
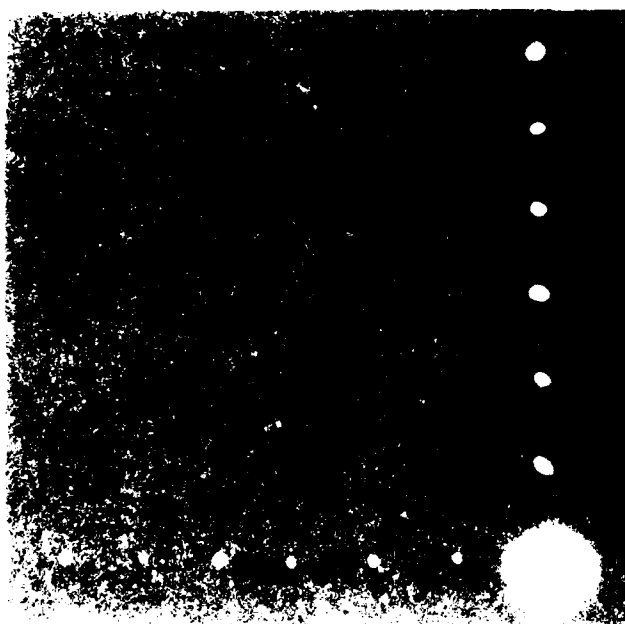
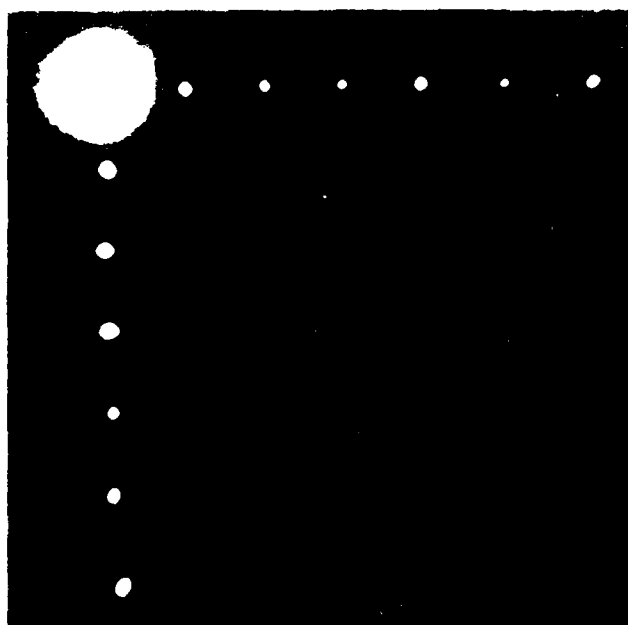
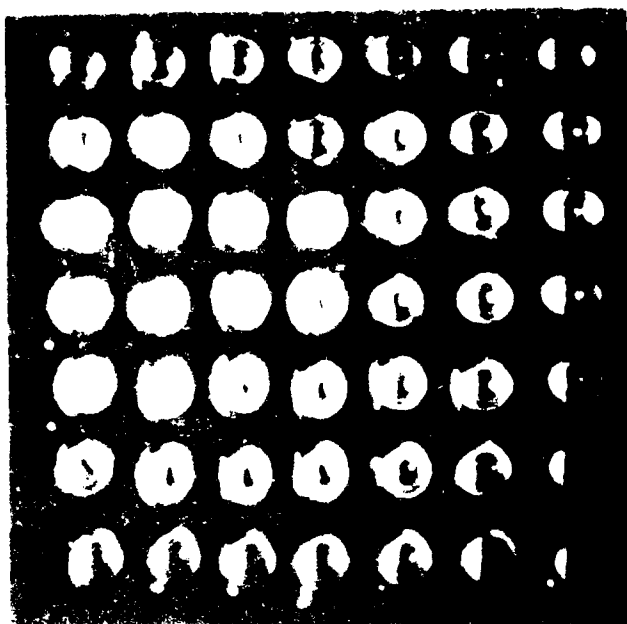


Figure 8.7 Mitsubishi ML4402 laser diode switching characteristics

For the next experiment, the board is completely populated with the 49 laser diodes and different rows and columns excited. Figures 8.8b, 8.8c and 8.8d show the output intensity spots in the absence of the fiber-optic connections (LUT geometry is shown in Figure 8.8a) when the (1,1), (4,5) and (7,7) rows and columns are excited. These figures clearly demonstrate the concept of the interlaced electrode LUT and show that only the cross-point laser diodes are in a lasing mode (i.e., strong intensity) whereas the remaining laser diodes are more-or-less in a LED, sub-threshold, mode (i.e., low intensity). Figures 8.9a and 8.9b show the responses of the 7 laser diodes of the first row and the 7 laser diodes of the first column respectively, when driven with a 250 MHz RZ waveform. As these data show, all lasers have a clean response to at least 250 MHz RZ. Note that there is about 20% variation in the output light level because of the variation in threshold level and current-power characteristics of the different laser diodes. In Figures 8.10a and 8.10b we show the responses of the 7 laser diodes of the sixth row and the 7 laser diodes of the sixth column. Once again we can see that all diodes respond to the drive waveform of 250 MHz RZ. Note, however, that the response is not as clean as that of Figure 8.9. Specifically, there is ringing and undershoot that increases with distance along the strip line. This problem is not well understood but is believed to be at least partially due to drive pulse reflections caused by imperfect termination of the strip line. This is a difficult problem to model and solve because of the dynamic impedances of the laser diodes involved and is compounded by the fact that we are dealing with laser diodes of different characteristics. It is important that this issue be further studied in detail to provide a solution which will eliminate the possibility of false responses by ringing.

Finally, we have measured the "noise" due to the response of the laser diodes which are connected to the rows and columns being exercised, but which are not located at the cross-points. This is an important



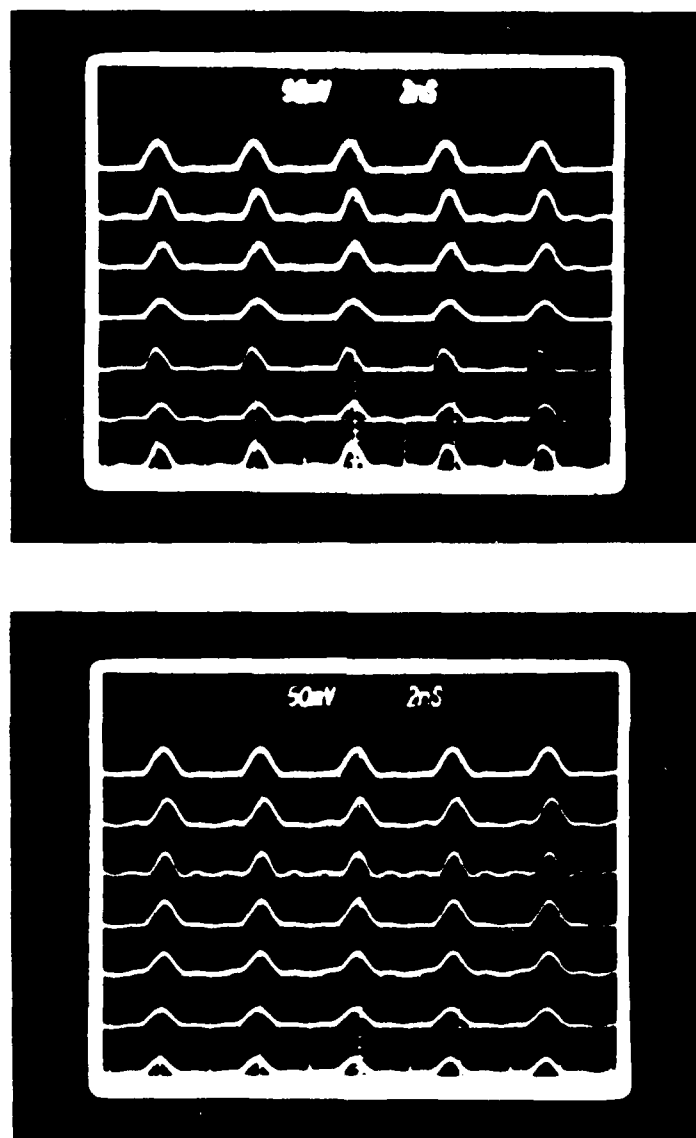


Figure 8.9 Responses of the 7 laser diodes of the first row (a) and of the 7 laser diodes of the first column (b).

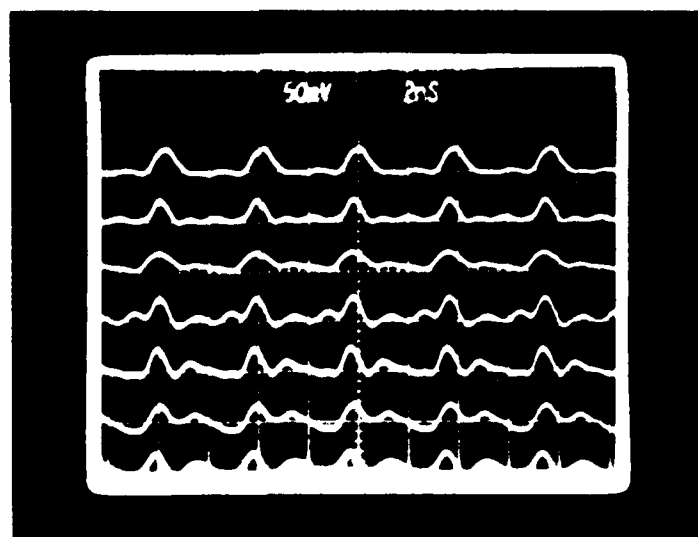
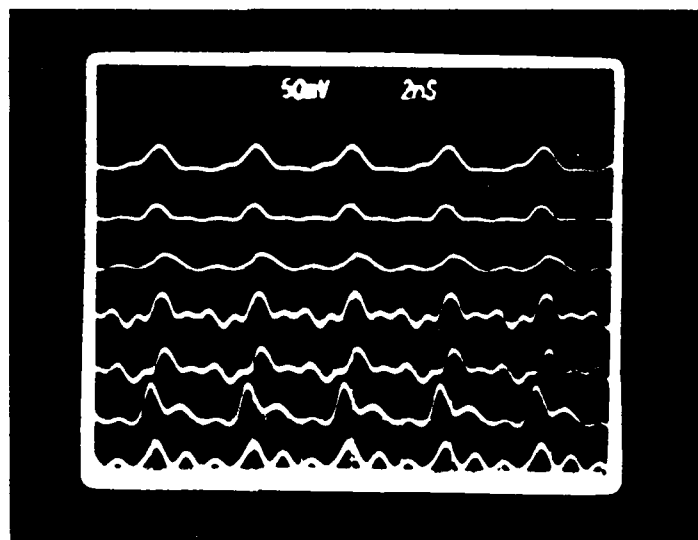


Figure 8.10 Responses of the 7 laser diodes of the sixth row (a) and of the laser diodes of the sixth column (b).

measurement because it provides information concerning the detector threshold level necessary in order to avoid false responses.

Figure 8.11 shows the responses of the 7 laser diodes of column 4 when the (4,3) laser diode is being exercised. As Figure 8.11 suggests the response is rather small, about 20 db below the response of the (4,3) laser diode, which translates to rather non-critical thresholds. Similar results are obtained when different rows and columns are excited and, thus, this is not a serious problem.

In conclusion, we have shown that a LUT constructed with discrete components can operate to at least 250 MHz RZ or 500 MHz NRZ data rates. We expect substantially higher rates from a hybrid or monolithic package. However, we should emphasize that a complete analysis of the driving circuit is necessary to enable properly terminated lines to be designed that will eliminate the ringing/undershoot problems.

8.5 Binary-to-Residue Conversion (B/R)

A simple way to understand B/R conversion is through a simple numerical example. Consider the binary representation of the number 255:

$$255 = 1 (2^7) + 1 (2^6) + \dots + 1 (2^0) = 11111111 \text{ (binary)}$$

Because of the presentation of algebraic rules in RNS (i.e., $\langle r+s \rangle = \langle r \rangle + \langle s \rangle$ and $\langle rs \rangle = \langle r \rangle \langle s \rangle$, where $\langle r \rangle$ is the residue representation of number r), the residue representation of 255 is equal to the residue representation of the sum of the products of the binary bits and the powers of 2 they correspond to. For example, the residue of 255 modulo 7 is

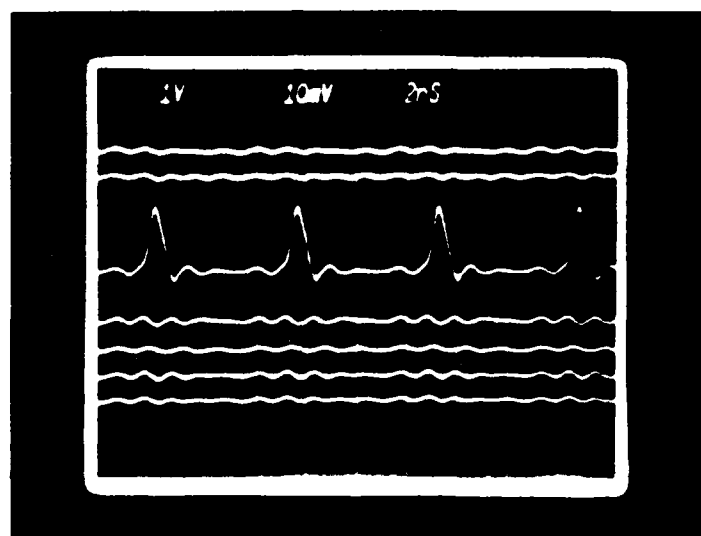


Figure 8.11 Responses of the 7 laser diodes of column 4, when the (4,3) laser diode is excited.

$$\langle 255 \rangle_7 = \langle 1 \cdot 2^7 \rangle_7 + \langle 1 \cdot 2^6 \rangle_7 + \dots + \langle 1 \cdot 2^0 \rangle_7$$

and the total prebias power

$$\begin{aligned} &= \langle 2 + 1 + 4 + 2 + 1 + 4 + 2 + 1 \rangle_7 \\ &= \langle 17 \rangle_7 = 3 \end{aligned} \tag{8.2}$$

From Equation (8.2) we see that for an efficient B/R conversion, we need to: (1) multiply each input bit by the residue representation of its corresponding power of 2 and (2) sum all the residues.

Fortunately, all these operations can be achieved in a pipelined architecture using the LUT technology of the previous section. An example of such an implementation is shown in Figure 8.12 for an 8-bit A/D converter and modulo 7. Each output bit of the A/D converter is connected to a binary switch. Pairs of binary switches are connected to 2x2 LUTs. Pairs of 2x2 LUTs are connected to 4x4 LUTs which in turn are connected to 7x7 LUTs. Each 2x2 LUT provides the residue of the sum of two bits. depending on whether each bit is "1" or "0", the binary switch activates one of the two possible electrodes (each bit can be 0 or Y in residue, Y being dependent on the power of 2 each bit corresponds to and the modulo we use). Once two inputs are present in the 2x2 LUT, one output (out of four possible outputs) is produced. For example, for the case of bits 2^7 and 2^6 , if we have $1:2^7$ and $1:2^6$ the LUT output is 3. Two such outputs now drive 4x4 LUTs whose outputs drive the 7x7 LUT. Note that this is a pipelined process and thus at each clock cycle a new residue is produced.

For most practical residue processors, moduli that exceed 11 are required. In these cases 2x2 and 4x4 LUTs are always used. The dimensions kxk of the third LUT depend on the modulo k we use. For example, for modulo 13 we need a 13x13 LUT. The number of 2x2, 4x4 and kxk LUTs we need depends on the number of input binary bits. Table 8.1 shows the number of 2x2, 4x4 and 7x7 LUTs we need for 8, 16 and 32 input bits when modulo 7 is used.

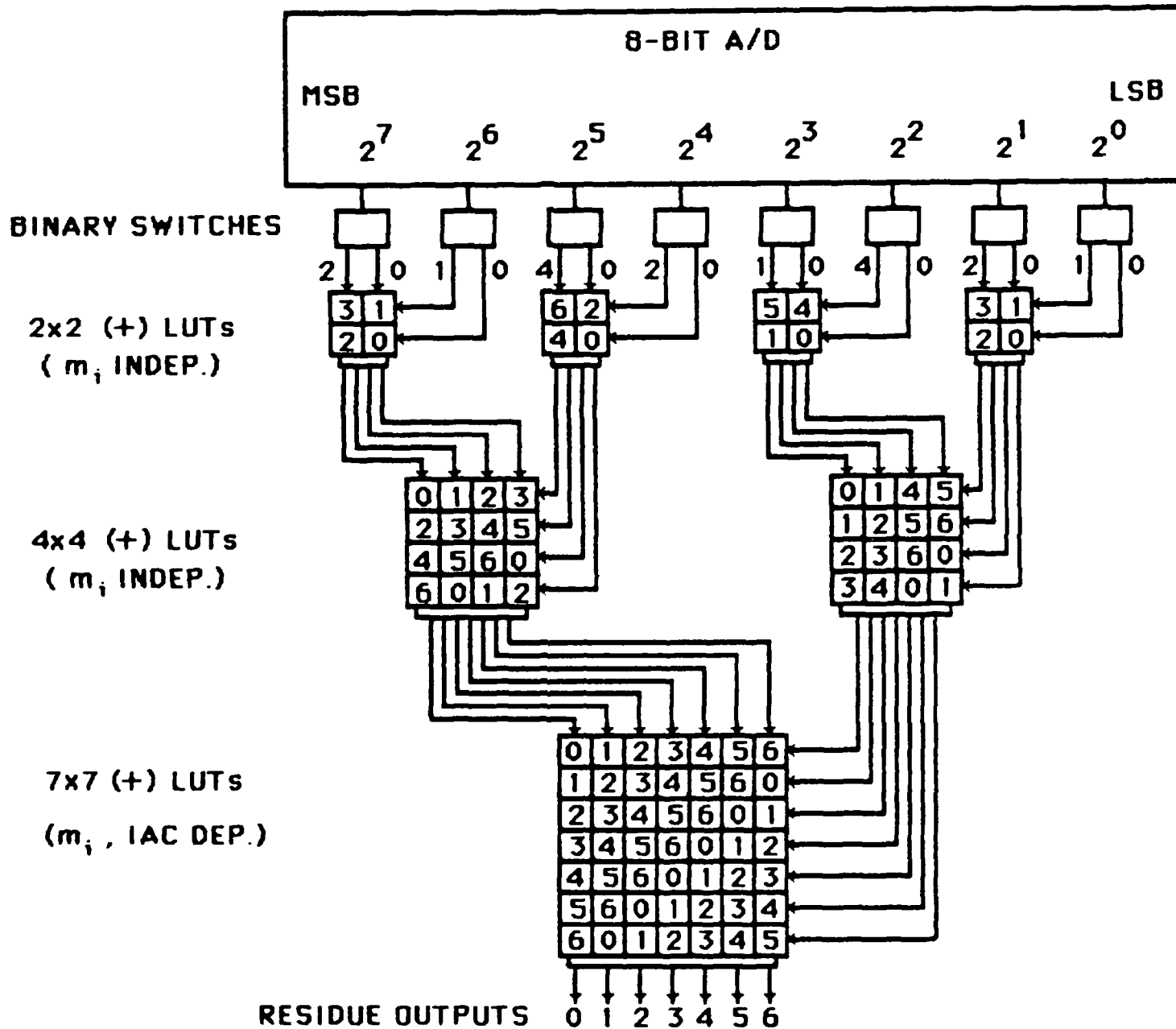


Figure 8.12 B/R Conversion via Pipelined LUTs

Table 8.1

N (bits)	2x2	4x4	7x7
8	4	2	1
16	8	4	3
32	16	8	7

From the above we can conclude that B/R conversion can be achieved in a very simple way by means of LUT technology. The process can be of high speed because a pipelined architecture can be used.

8.6 Residue-to-Binary Conversion (R/B)

Residue-to-binary conversion can be achieved by converting into the mixed radix system¹⁹ which allows for the use of LUTs.

The principles of operation are pictured in Figure 8.13 for a system of four moduli, m_1 through m_4 , with residues r_1 through r_4 . The four residues are clocked in simultaneously with r_2 , r_3 and r_4 going to LUT accumulators and r_1 being fed through an additive inverter (for complement calculation) to each LUT accumulator; r_1 also passes through the system as output a_1 .

The subtractions are performed to the respective moduli and the results are passed to the LUTs. There they are modulo multiplied by r_1 and passed on to the next stage. The second stage performs similarly to the first except that now a_2 is subtracted from the others and is passed to the output. The process continues through a cascade until the final output is triggered, in this case by the arrival of a_4 . The

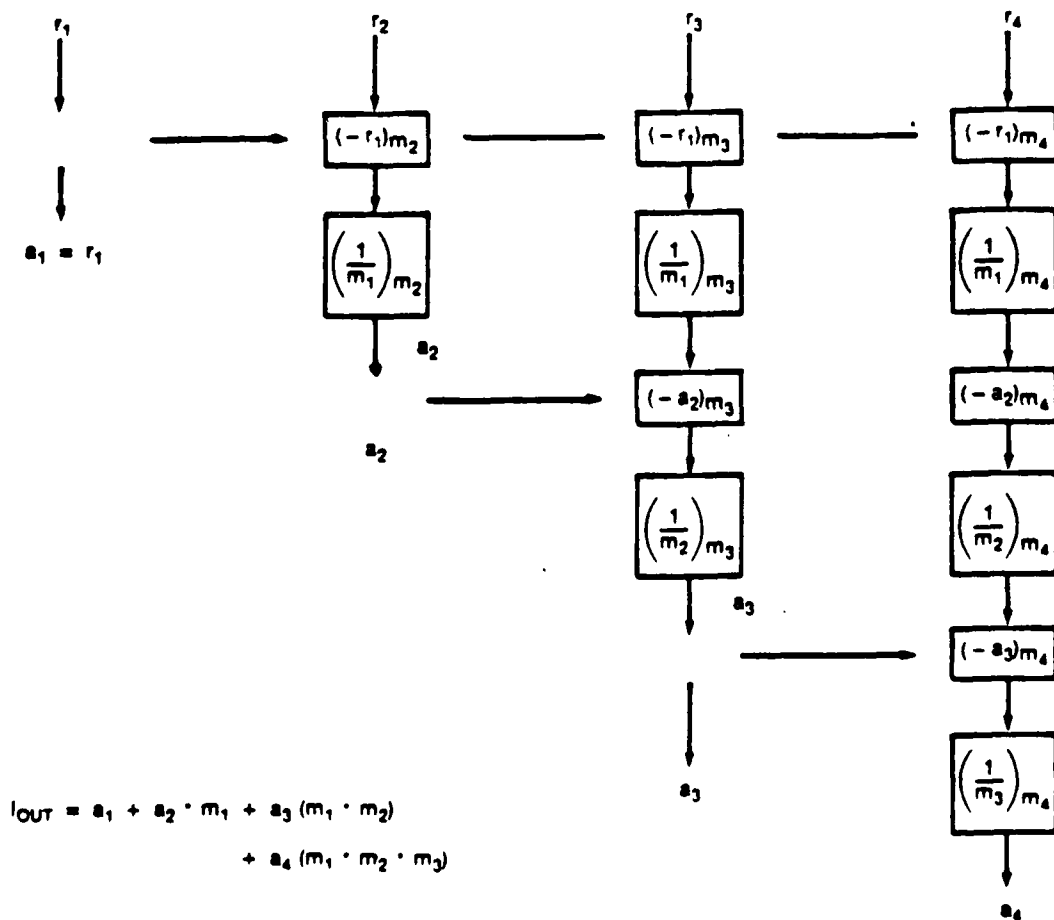


Figure 8.13 Schematic Operation of 4-Residue Mixed Radix Decoder for Residue to Integer Transformation [after Tai et al. (Ref. 21)].

result of the decoding process must then be calculated according to the expression

$$I = a_1 + a_2 \cdot m_1 + a_3 \cdot m_1 m_2 + a_4 \cdot m_1 \cdot m_2 \cdot m_3 + \dots \quad (8.3)$$

where I denotes that the result is an integer and the sequence continues for as many terms as there are residues.

An example of this procedure is shown in Figure 8.14. In this example we use 4 moduli 2,5,7 and 9. The input residue representation is (1,3,4,0) which corresponds to 333. Rectangular blocks show the LUTs and small squares the output detectors. The values within the detector blocks correspond to the results of the wiring maps, for the additive inverters $(-r_i)_{m_j}$, when applied to the original input values (shown next to the arrows). One can understand the operation by tracing the heavily outlined squares in each block. In the bottom of Figure 8.14 we show the conversion result which is 333.

Tai et al.²¹ have suggested a pipelined version of the R/B converter. In their approach the coefficients a_1 , a_2 , a_3 , etc., are delayed so that they all appear at the same time. Figure 8.15 shows the pipelined version of the R/B converter of Figure 8.14. Note that delays are implemented using technology similar to that used in LUTs; i.e., sets of laser diodes, fibers and detectors. Through this pipelining structure, we can have multiple sets of residues following each other through the R/B converter with a time-gap of only one cycle and thus high speed conversions are possible. Note that the total number of LUTs is $2N-2$ where N is the number of moduli we use.

Let us consider now a practical implementation of Equation 8.3, assuming that a pipelined R/B decoder is available. The first operation we need to do is to multiply each coefficient a_i by the

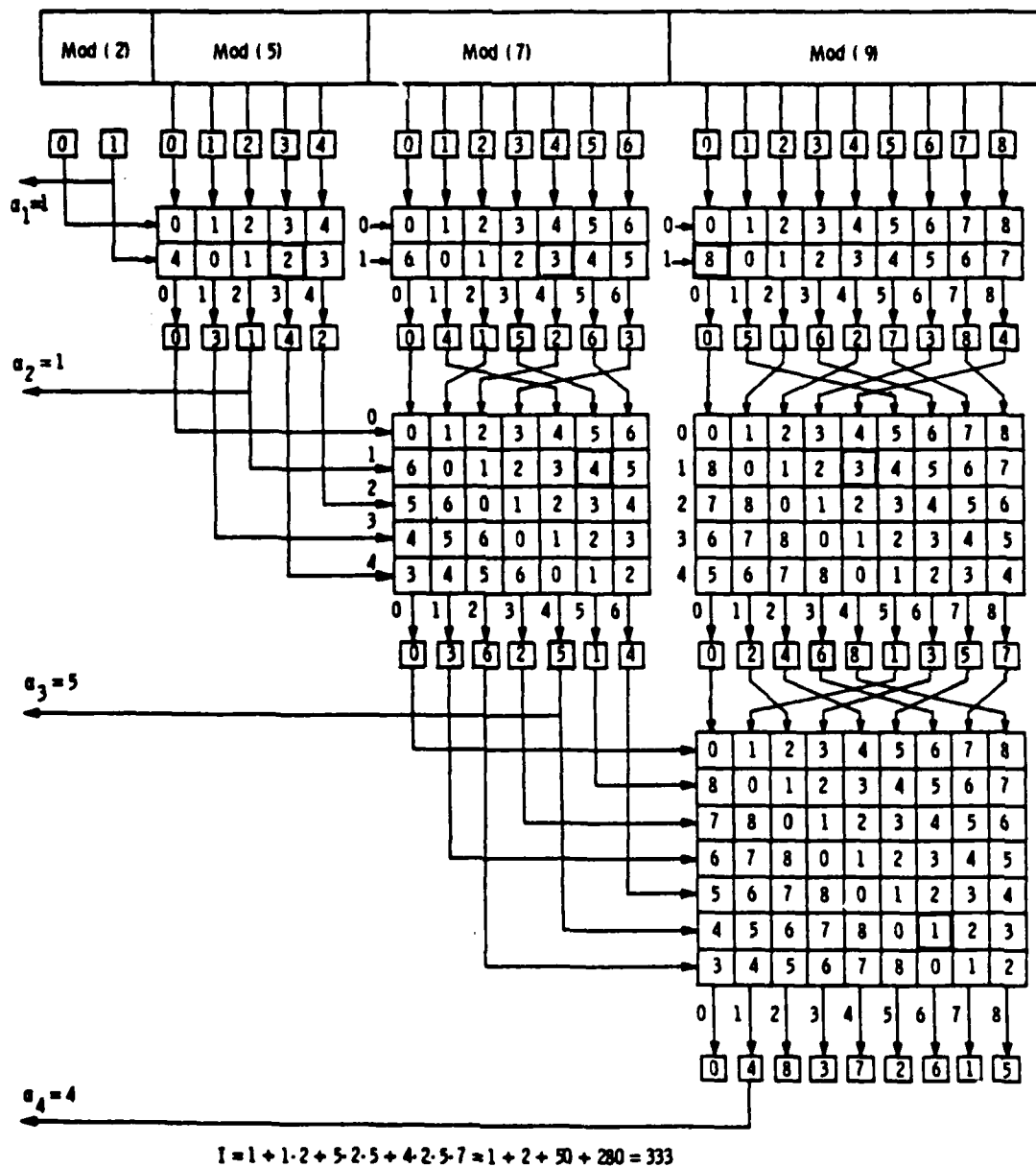


Figure 8.14 R/B conversion example

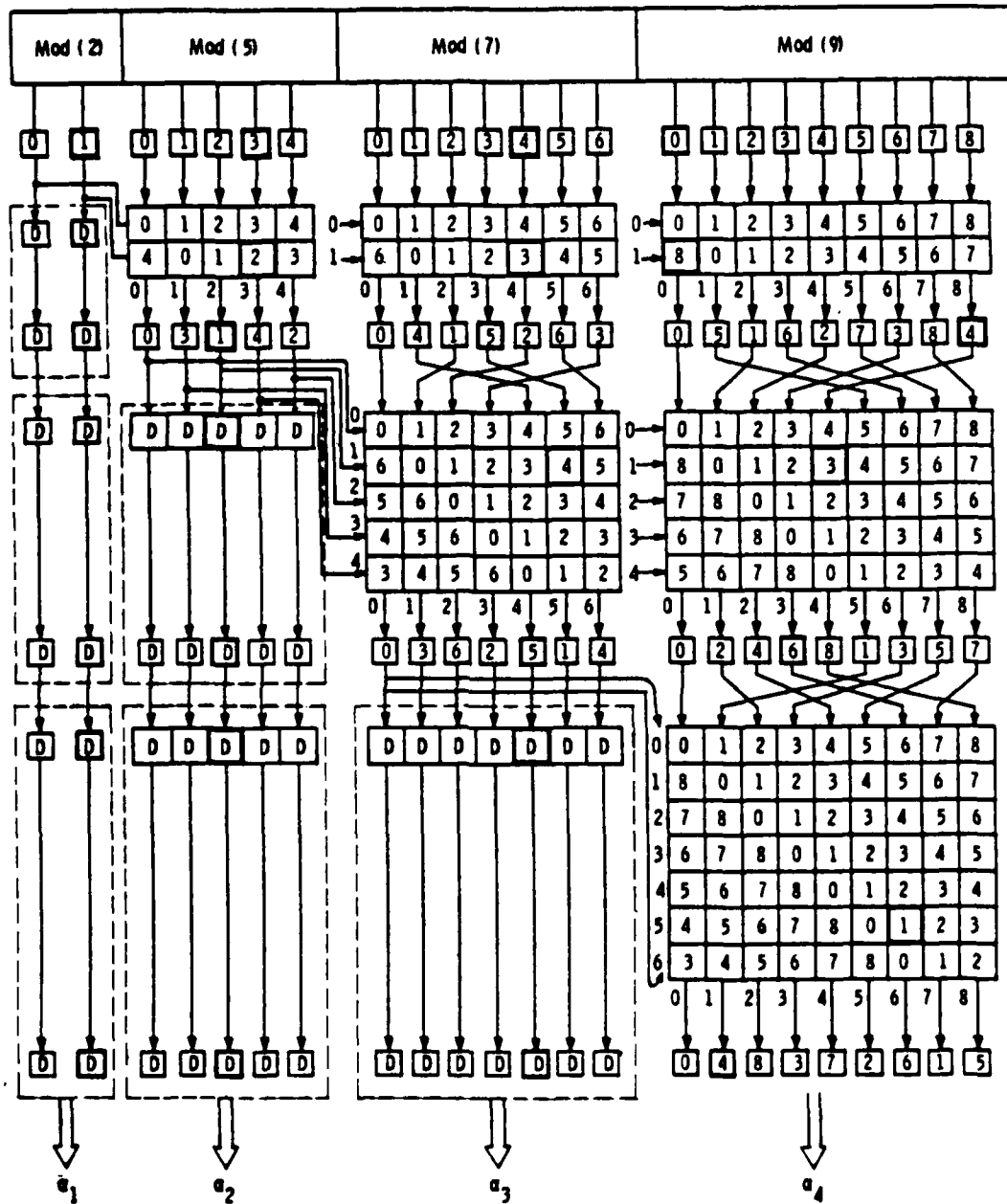


Figure 8.15 R/B pipelined conversion example

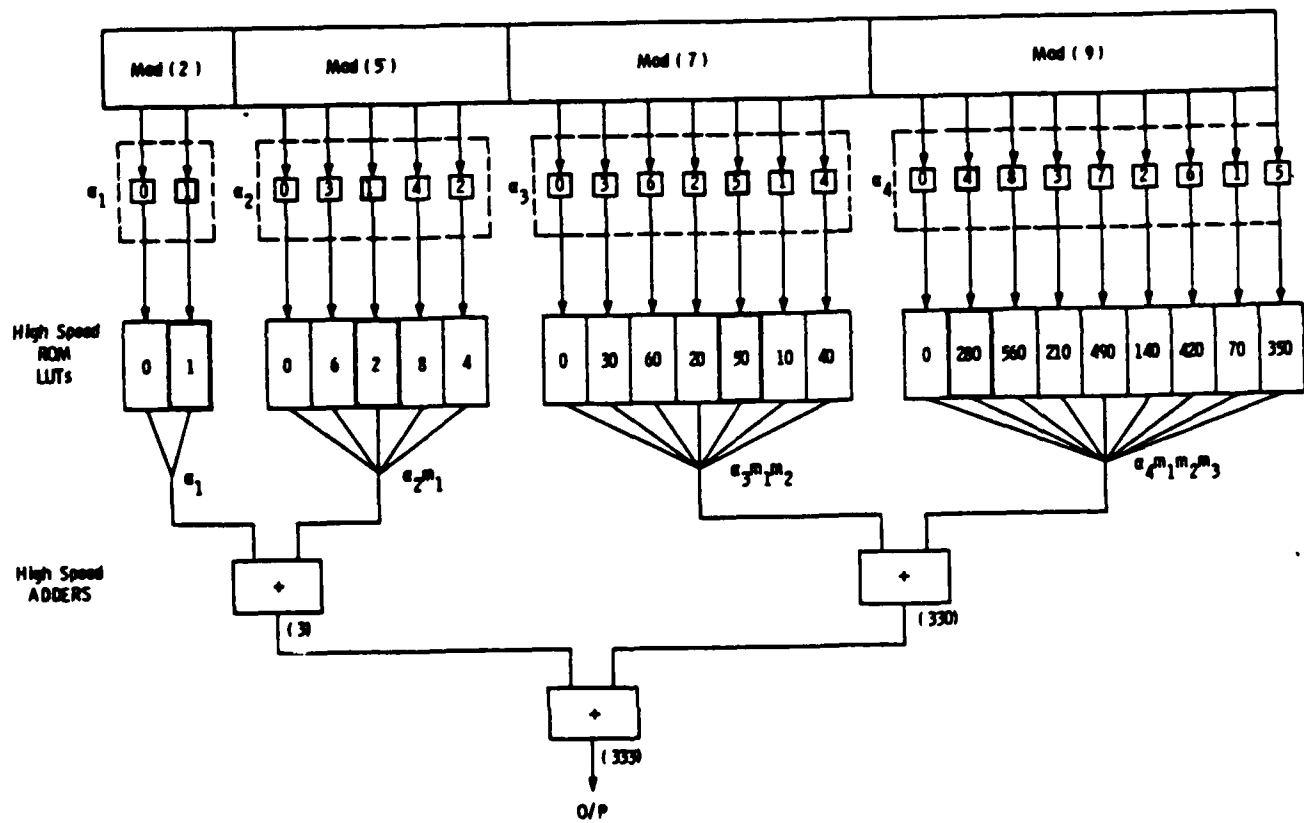


Figure 8.16 Pipelined LUT implementation of Eq. 8.3

proper m_i, m_j, \dots, m_w product (m_i, m_j, \dots, m_w products are known a priori). This operation can be accomplished via high speed digital ROM LUTs (see Figure 8.16), which are activated depending on which value of the a_i coefficients is "on." Next, we fan-in all outputs of each a_i ROM LUT and add them in pairs via a high-speed digital adder. Subsequently, we sum the results from the adders to obtain the final output I . In Figure 8.16 we present an example of this operation for the R/B converter of Figure 8.15. It is important to note that the approach of Figure 8.16 is fully pipelined which means high speed conversion capability. Note that the total number of ROM LUTs we need is equal to the sum of the N moduli we use, whereas the total number of high-speed adders is equal to $N-1$.

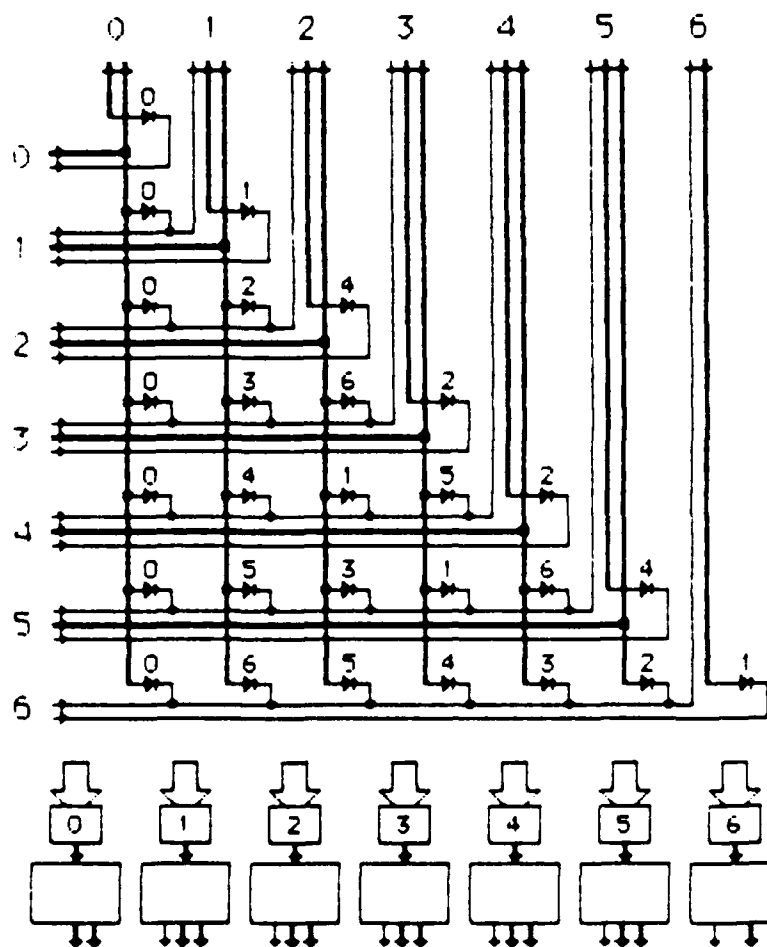
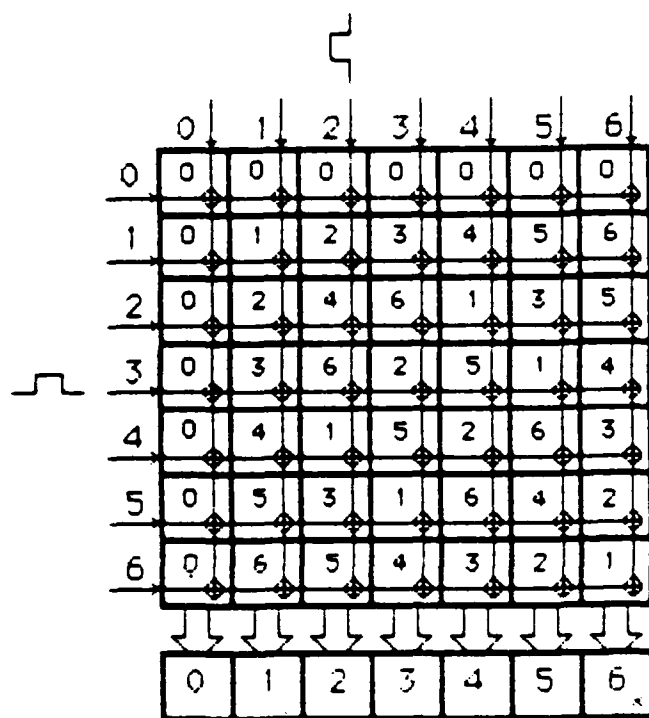
From the above we see that R/B conversion can be achieved in a very simple pipelined way with efficient utilization of LUT technology.

8.7 Hardware Minimization

One of the issues associated with position-coded LUTs is their complexity in terms of numbers of gates (optical sources in our cases). This is because the number of gates N_g grows as the square of the modulo m . To understand this, consider a particular residue example, specifically, a residue equivalent of an 8-bit multiplication, which requires moduli 3, 5, 7, 11 and 13. In this case, we need a total of 437 gates or laser diodes. For computationally linear signal processing problems this is more-or-less acceptable but for computationally non-linear applications it becomes a serious limitation. For example, consider the dynamic range needed in order to solve a system of linear equations of dimension 12. Assuming that the maximum value of the determinant is 128, we find (see Section 9.2, Equation 9.5) that the dynamic range needed is of the order of 5×10^{31} (about 105 binary bits)! To accomplish this, we need the moduli 9, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41, 43, 47, 49, 53, 59, 61, 64, 67, 71, 73. In this case just one residue multiplier requires a total of 42,452 laser diodes.

Obviously, for this kind of problem the complexity grows rapidly. In an effort to minimize this problem, we have sought solutions at both the LUT and processor levels. In the first part of this section, we discuss our attempt to minimize the number of laser diodes per LUT and in the second part we discuss a general residue scaling approach suggested by J. N. Polky et al.²²

One can accept that the LUT gates can be reduced once it is realized that there exist symmetry planes in both multiplier and adder LUTs. Consider, for example, the LUTs of Figure 8.3. In the multiplier LUT one can see that the symmetry line passes from (0,0) to (6,6) coordinates. Similarly, the adder LUT is symmetric about the line from (0,6) and (6,0). Thus, if we route some of the inputs properly, about one-half of the diodes will be needed. One such example of a reduced LUT is shown in Figure 8.17 for the case of a modulo 7 multiplier LUT. It can be seen that we need 28 diodes versus 49 for the general implementation. This is achieved via proper interconnection of similar output channels. In the process, however, we are forced to use multiple-drivers per grid, whereas for the non-reduced case we need only single drivers. In addition, we have increased the average number of interconnections (per diode) by a factor of 2. Thus, we see that the reduction of gates is accompanied by an increased complexity in the interconnect pattern. In fact, one can easily show that as the number of diodes decreases, the number of interconnections per diode increases. This relationship is shown in Figure 8.18 where we plot the number of laser diode rows (N_R) versus the number of interconnections per diode (n_I) for modulo m_c . From this plot we can see that, in principle, a LUT made out of 1 row of gates (total of m_c diodes) is possible; however, each diode needs at least $2 m_c$ interconnections. Thus, we need to optimize the relative numbers of diodes and interconnections; one method of accomplishing this is to assign cost functions to both diodes and interconnections. For our experimental LUT boards a simple analysis shows that the multiple drivers and the



REDUCED IMPLEMENTATION

NOTATION

- ✚ - LASER DIODE OR LED
- ➔ - POSITIVE DRIVE PULSE
- ➞ - NEGATIVE DRIVE PULSE
- ⌋ - MULTIPLE FIBER-OPTIC CONNECTION
- 4 - L^FTECTOR
- ☐ - MULTIPLE DRIVER

Figure 8.17 Examples of Reduced Size LUT via the use of symmetry

Curve 752092-A

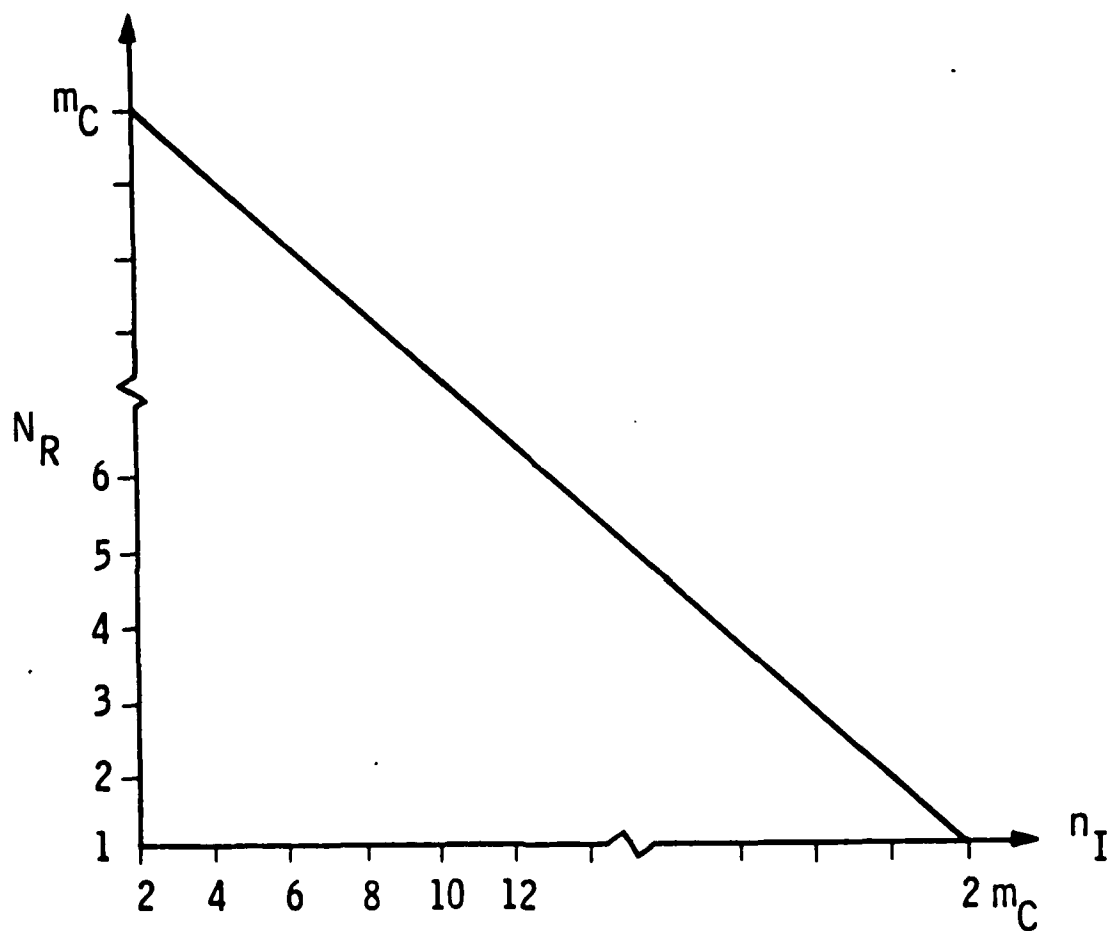


Figure 8.18 Number of Laser Diode Rows (N_R) versus Number of Interconnections per Diode (n_i) for Modulo m_C

more complicated interconnection pattern result in a more expensive and more complicated LUT and, thus, the unreduced LUTs are preferable. Note, however, that this analysis is performed for a discrete component LUT where the physical length of the electrode microstrip lines is of the order of 10 cm. In a realistic scenario where LUTs are made out of integrated LED or LD arrays and the microstrip lengths are considerably smaller (~ 1 cm), a similar analysis may yield different results. In any event our present feeling is that a reduction by a factor of 2 is about the limit of this approach. The result of such a reduction is obviously not very significant and, thus, we need to develop LUT architectures in which N_g grows at a slower rate than m^2 .

Hardware reduction can also be accomplished at the processor level by using techniques which include efficient algorithm selection (to keep the low dynamic range) and scaling.¹⁹ The choice of algorithm is application dependent. For the APAR case the only present alternatives are Gram-Schmidt type algorithms and possibly Gauss elimination techniques since these do not require divisions or square roots (which are problematic for RNS). Scaling is possibly a more profitable approach since it can be applied to a number of applications. The scaling technique we have studied is that suggested by J. N. Polky et al²² which offers the advantages of: (1) scaling for both positive and negative numbers and (2) pipelining. In this scaling technique²² we first prescale by adding $M/2$, next scale by performing about $N+1$ additions per module, and then post scale by subtracting various biases. The technique requires that an extra module is used. For the prescaling, one wiring map per module is required and its output is fanned out to $N+1$ channels. For the scaling operation, each module requires N maps whose results need to be added and, thus, $N-1$ adder LUTs (per module) are needed together with one more adder LUT for a combined scaling/post-scaling calculation. Thus, the total number of LUTs needed is $N^2 + N$. It is of interest to examine the accuracy and range of this scaling procedure so that the "savings" (from the

scaling) and "expenses" (due to $N^2 + N$ LUTs) can be compared. To achieve this we use a computer program which calculates (in residue) the scaling results for a variety of scale factors S and residue dynamic ranges M . In Tables 8.2, 8.3 and 8.4, we show the worst-case scaling accuracy (SA) and the expected accuracy (EA, obtained by performing a straightforward division of the number to be scaled by S) for different residue ranges M ($M = 27, 24$ and 20 binary bits respectively) and scaling factors S . For each case, 512-65,000 numbers are scaled ($S = 0.5\% - 0.0001\%$ of M) and the lowest SA is reported. From the above tables we see that SA varies as a function of both M and S , with the latter being the most critical. This implies that for a given scaling accuracy, the scaling factor cannot exceed a given percentage of M . For example, for $M = 24$ (Table 8.3) if a SA of 9 bits is required, then S cannot exceed 0.05% of M .

From Tables 8.2-8.4 we see that if a SA of 9-10 bits is desired, the maximum scaling factor S cannot exceed 0.005% M . This implies that, in principle, with M values of about 20,000 we can handle much larger dynamic range problems. This issue needs to be studied further in conjunction with a specific algorithm. Only in such a context can we evaluate whether the benefits of scaling are significant. Unfortunately, the Gram-Schmidt algorithm of Section 9 cannot incorporate this scaling technique.

8.8 System Characteristics for a Square Systolic Residue System

In this section we discuss an example of a residue LUT processor, namely, a systolic processor¹⁵ for matrix-matrix multiplication. This particular example is chosen because a large number of APAR algorithms can be expressed in terms of matrix-matrix multiplication. Figure 8.19 shows a typical configuration for a square systolic array. The array consists of $n \times n$ MAU LUTs, each of which consists of cascaded multiplier

Table 8.2

S	% M	EA (Bits)	SA (Bits)
200,000	0.5	9	7
100,000	0.1	10	7
50,000	0.05	11	9
25,000	0.01	12	9
12,500	0.005	13	9
6,250	0.001	14	12
3,125	0.0005	15	12
1,562	0.0001	16	13

Table 8.3

S	% M	EA (Bits)	SA (Bits)
41,500	0.5	9	7
20,750	0.1	10	8
10,370	0.05	11	9
5,190	0.01	12	9
2,600	0.005	13	10
1,300	0.001	14	10
650	0.0005	15	11
320	0.0001	16	13

Table 8.4

S	% M	EA (Bits)	SA (Bits)
1800	0.5	9	7
900	0.1	10	8
450	0.05	11	9
230	0.01	12	9
110	0.005	13	11
60	0.001	14	12
30	0.0005	15	13
10	0.0001	16	13

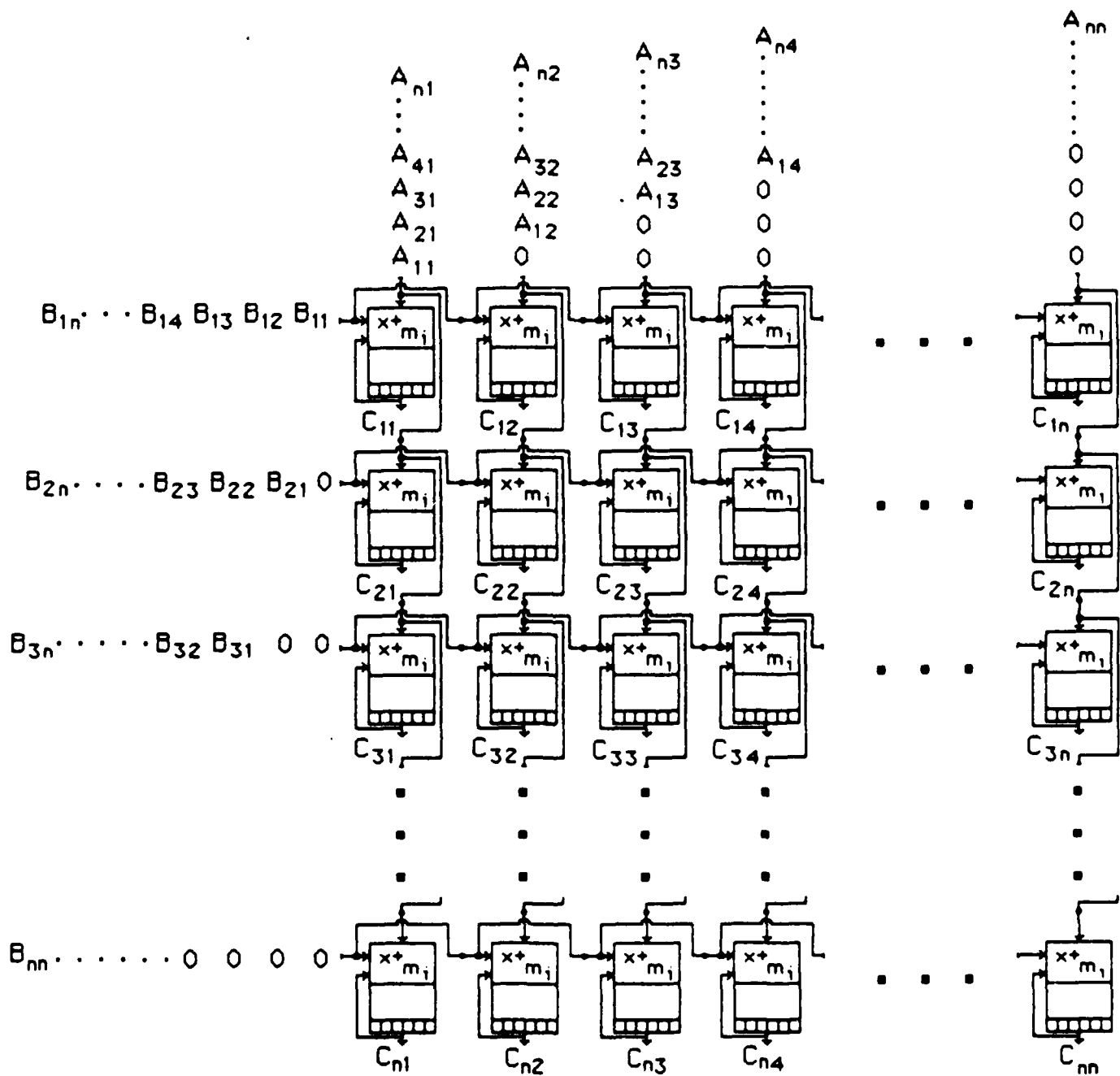


Figure 8.19 LUT Implementation of a Square Systolic Array for Matrix-Matrix Multiplication

and adder LUTs. To insure proper data propagation, local interconnects between adjacent MAU LUTs involve a fixed delay which is equal to the delay of a LUT. For the purposes of this example, we assume a 16×16 real matrix multiplication scenario in which the elements of the input matrices cannot exceed 8 bits. Each MAU is required to add 16 products each consisting of 16 bits. Thus, the dynamic range requirement is 20 bits or 1,048,576. To handle this we use the moduli 7,8,9,11,13 and 17 which yield an M of 1,225,224. Note that in this example the dynamic range of the multiplier can be handled by moduli 7,8,9,11 and 13; however, we forced to use the extra modulo 17 because of the dynamic range requirement of the adder. For the systolic processor we need $2 \times 16 \times 16 = 512$ LUTs per modulo layer giving a total number of $512 \times 6 = 3,072$ LUTs. The total number of LDs in these LUTs is about 396,000. Additional LUTs are required for B/R and R/B conversion. We need a total of $6 \times 16 \times 2 = 192$ B/R converters each consisting of 7 LUTs. The average number of LDs per B/R is about 170 and, thus, the total number of LDs in the B/R converters is about 32,000 (i.e., about 8% of the number if the processor). The number of R/B converters is determined by the read-out arrangement. Let us assume the use of 16 R/Bs in order to read out the results in a pipelined fashion. Each R/B needs 15 LUTs which contain about 2,700 LDs and thus the total number of LDs for the R/Bs is about 43,000 (i.e., about 11% of the number in the processor). Thus, we see that the total number of LDs is about 470,000. By comparison, a fully digital 16×16 array processor requires a total of 256 MAUs. Each MAU requires about 76 Full Adders each of which requires about 10 gates. Thus, the total number of gates is 190,000 which is about 40% of the complexity (in terms of gates) of the residue system. Thus, we conclude that, for this particular square systolic processor, the residue LUT implementation has twice the complexity of an electronic digital implementation. However, this situation can change if the complexity of the LUT is reduced. We also note that the specific application which we have analyzed does not favor the residue system. This is because the multiplier LUTs have an

increased complexity dictated by the dynamic range of adder LUTs which requires the additional modulo 17 LUTs. Without these LUTs the number of gates in the processor is reduced to about 270,000, comparable to that of a fully electronic processor. This emphasizes the necessity of choosing applications which are well suited to residue implementation.

The MS of the system is equal to that of the LUTs which we expect to be in the range 1-6 GHz. To calculate the SE of the processor we perform an analysis similar to the one in Section 8.2. Thus, assuming 1 GHz operation and LUTs implemented with laser diodes, a SE of the order of $2-3 \times 10^9$ M-A/sec. W can be expected. These values of MS and SE for the residue processor are superior by at least an order of magnitude to those of a GaAs implemented electronic processor.

The performance of the residue processor may be improved considerably if different Electro-Optic technology, for example SEED (Self-Electro-Optic-Effect) devices,²⁶ is developed for LUT implementation. These devices exhibit strong changes in optical absorption (transmission) dependent on the intensity of the incident light. These changes are due to changes in internal electric field distribution that occur in response to a variation in carrier concentration induced by optical absorption. The response time of this process is estimated²⁶ to be as short as 2×10^{-13} sec. If this technology can be implemented in LUTs then improvements in MS and SE by 2 orders of magnitude can be expected. In this case, the performance of the residue processor would be far superior to that attainable from electronic processors, including GaAs.

9. RESIDUE LUT IMPLEMENTATION OF GRAM-SCHMIDT APPROACH FOR THE SOLUTION OF LINEAR SYSTEMS

9.1 Introduction

In the previous chapter, we showed how residue LUT techniques can be used to perform simple arithmetic operations. In this chapter we discuss the residue LUT implementation of a variant of the Gram-Schmidt¹ approach for the solution of systems of linear equations. As such this technique is directly applicable to the APAR problem. In the next Section 9.2, we discuss the problem of solving systems of linear equations using residue arithmetic. In Section 9.3 we present the basics of the Gram-Schmidt variant along with a numerical example. Section 9.4 contains the LUT design for the residue implementation of the technique. Finally, in Section 9.5, we discuss the characteristics of the LUT processor.

9.2 Residue Resolutions of Linear Systems

A significant advantage of RNS involves its considerably greater flexibility than DMAC or HPAM. Addition, subtraction, multiplication and some forms of division can be performed in easily-implemented LUTs, and this allows the efficient implementation of rather complicated signal processing algorithms, such as the Gram-Schmidt¹ orthogonalisation approach, in solution of large linear systems of equations.

These considerations strongly recommend the exploitation of RNS in the APAR application, which in certain formulations requires solutions of systems of the form

$$C_{v \leftarrow v} w = s_v \quad (9.1)$$

where C_v is the covariance matrix, w_v is the adaptive weight vector, and s_v is the steering vector. Adoption of RNS to solve Eq. (9.1) precludes consideration of the QR^1 and certain other often-used algorithms which are inherently tied to real, as opposed to integer, calculations. This is not the case, however, with the modified version of the Gram-Schmidt orthogonalization, as we shall illustrate in detail in later sections. Before we address these particular algorithms, it is appropriate to look more generally at the use of residue arithmetic in solving sets of linear equations.

To begin with, we assume that the elements of C_v and s_v are all integers; solutions in RNS can also be computed for the more general case where data are given as Gaussian integers, but we avoid this formulation for simplicity. Because of the manner in which the covariance matrix is derived from noisy signals, it is relatively safe to assume that C_v is non-singular. With C_v^{adj} representing the adjoint of C_v , we know that

$$y = C_v^{adj} s_v \quad (9.2)$$

is the integer solution of

$$C_v y = (\det C_v) s_v \quad (9.3)$$

Thus, we can write

$$w_v = y / \det C_v \quad (9.4)$$

which shows that division can be postponed until the last step of computation. Newman²⁷ has proved that Eq. (9.3) can be solved using residue arithmetic and we highlight some important steps of his proof here. Let Z be the ring of all integers and, for a given integer $k > 0$,

let Z_k represent all integer multiples of k . Basically, use is made of the ring Z_M of integers modulo M ; it is the ring of residue classes of the form $r + Z M = : \langle r \rangle_M$. Then $Z_M = \{\langle 0 \rangle_M, \langle 1 \rangle_M, \dots, \langle M-1 \rangle_M\}$ and the well-defined rules $\langle r \rangle_M T \langle s \rangle_M = \langle rTs \rangle_M$ with $T = +$ or \times , make Z_M a commutative ring. These rules justify computations involving finite sums and products; for example, if B is a square matrix, then $\langle \det B \rangle_M = \det \langle B \rangle_M$. In solving linear systems of algebraic equations, Newman has shown that the principal modulus M must bound certain parameters involving the determinant of C_v and the second member \underline{s}_v . Using Hadamard's inequality and Cramer's rule we have derived the following slightly improved inequality for a bound on M :

$$M > 2^{n/2} K^{n-1} \max \{k, b\} \quad (9.5)$$

where $K = \max_{i,j} |c_{v_{ij}}|$ and $b = \max_j |s_{v_j}|$. In the following section examples are presented which show that this bound is conservatively large, although improving it is difficult.

9.3 Gram-Schmidt Variant

The Gram-Schmidt approach is applicable to the APAR problem because it allows operations involving rectangular data matrices A , where $C_v = A^* A$. Here, we write

$$A^* A \underline{w}_v = \underline{s}_v \quad (9.6)$$

which we solve for \underline{w}_v . This is done by post-multiplying A by a sequence of $n \times n$ matrices to produce a new sequence of $m \times n$ matrices where each succeeding matrix in the latter sequence has its number of orthogonal columns increased by 1. With E_1 representing the $n \times n$ identity matrix,

$$AE_1 = Q_1. \quad (9.7)$$

If E_2 orthogonalizes the second column of AE_1 to its first, then

$$AE_1E_2 = Q_2 \quad (9.8)$$

has two orthogonal columns. This sequence continues until we have $E = E_1 \dots E_n$ and $Q = Q_1 \dots Q_n$, where E is upper triangular because all of its constituents are used and Q has orthogonal columns, and, consequently, Q^*Q (Q^* is the Hermitian conjugate of Q) is a diagonal matrix we call Λ . This procedure differs slightly from the usual Gram-Schmidt method in that we use orthogonalization without normalizing the vectors in order to obviate the excessive growth of integers in the computation as well as to postpone division until the last step and remove the necessity to extract roots. The matrix E_{k+1} differs, at most, from the identity by possessing integers in the upper half of its $(k+1)^{st}$ column. In summary, we now have

$$AE = Q \quad (9.9)$$

and multiplying this equation on the left by its Hermitian conjugate gives

$$E^*A^*AE = Q^*Q = \Lambda, \quad (9.10)$$

$$A^*A = E^{*-1}AE^{-1} \quad (9.11)$$

and

$$\underline{w}_v = EA^{-1}E^*\underline{s}_v. \quad (9.12)$$

To perform this computation in residue arithmetic we must also evaluate the determinant of C_v in each modulus. From Eq. (9.11) we see that this number is given by

$$\det C_v = \det A / |\det E|^2 \quad (9.13)$$

where only the products of diagonal elements in E and, of course in A need be computed. These computations are illustrated with an example. Before proceeding with this example, we should note that the growth of integers in this algorithm can be enormous but that it is not the individual collection of integers appearing in the intermediate steps of the computation that must be bounded by the principal modulus -- instead, we postulate that it is the bound on the coefficients in the expansions of various terms that matters.

Consider now a 4x3 system corresponding to Eq. (9.1) and given by

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ s \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \quad (9.14)$$

$$A^* \quad A \quad \underline{w}_v = \underline{s}_v$$

with $\underline{w}_v = [x, y, s]^t$. As we described above, the columns of A are orthogonalized through application of the matrix

$$E = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix} \quad (9.15)$$

to produce

$$Q = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 3 \\ 1 & -1 & 2 \\ 1 & 1 & -1 \end{bmatrix} \quad (9.16)$$

and

$$A = \begin{bmatrix} 3 & & \\ & 3 & \\ & & 15 \end{bmatrix} \quad (9.17)$$

Using Eq. (9.13) we find that

$$\det C_v = 15 \quad (9.18)$$

and to insure that an integer solution exists, we multiply $\underline{s}_v = (1,1,-1)^t$ by 15 and solve for $\underline{w}'_v = (x',y',z')^t$. The inverse of C_v is

$$C_v^{-1} = EA^{-1}E^* = \frac{1}{15} \begin{bmatrix} 11 & -4 & -3 \\ -5 & 5 & 0 \\ -3 & 0 & 9 \end{bmatrix} \quad (9.19)$$

so that we get

$$y = C_v^{-1} \begin{bmatrix} 15 \\ 15 \\ -15 \end{bmatrix} = \begin{bmatrix} 9 \\ 0 \\ -12 \end{bmatrix} \quad (9.20)$$

which by Eq. (9.4) yields $\underline{w}_v = (3/5, 0, -4/5)^t$. Note that C_v^{-1} is symmetric which is expected because C_v , the covariance matrix, is symmetric.

Let us now course through the same examples using residue arithmetic. From Eq. (9.5) we find that M should be ≥ 2808 , which can be satisfied with moduli of 5, 7, 8 and 11; in this particular example, using 2 or 3 as a modulus would cause $\langle E \rangle$ to be singular, a circumstance which cannot be tolerated. We have noted previously that in some cases smaller M values are sufficient for providing the correct answer, but this is not always true and one thus has to select an M at least equal to the upper bound. To demonstrate that smaller principal moduli can suffice we choose to solve our 4 x 3 example using only the two moduli 7 and 11. In modulo 7 we get

$$\langle E \rangle_7 = \begin{bmatrix} 1 & 6 & 6 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad (9.21)$$

$$\langle Q \rangle_7 = \begin{bmatrix} 1 & 0 & 6 \\ 0 & 1 & 3 \\ 1 & 6 & 2 \\ 1 & 1 & 6 \end{bmatrix} \quad (9.22)$$

$$\langle A \rangle_7 = \begin{bmatrix} 3 & & \\ & 3 & \\ & & 1 \end{bmatrix} \quad (9.23)$$

and

$$\langle \Lambda^{-1} \rangle_7 = \begin{pmatrix} 5 & & \\ & 5 & \\ & & 1 \end{pmatrix} \quad (9.24)$$

Thus, we find that

$$\langle C_v^{-1} \rangle_7 = \langle E \rangle_7 \langle \Lambda^{-1} \rangle_7 \langle E^* \rangle_7 = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 5 & 0 \\ 4 & 0 & 2 \end{pmatrix} \quad (9.25)$$

and

$$\langle Y \rangle_7 = \langle C_v^{-1} \rangle_7 \begin{pmatrix} 1 \\ 1 \\ 6 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix} \quad (9.26)$$

For modulo 11 we obtain

$$\langle Y \rangle_{11} = \begin{pmatrix} 0 & 7 & 2 \\ 7 & 4 & 0 \\ 2 & 0 & 5 \end{pmatrix} \begin{pmatrix} 4 \\ 4 \\ 7 \end{pmatrix} = \begin{pmatrix} 9 \\ 0 \\ 10 \end{pmatrix} \quad (9.27)$$

We can decode the results of Eqs. (9.26) and (9.27) to obtain

$$\begin{aligned}
x^1 &= (\langle 1 \rangle_7, \langle 9 \rangle_{11}) = \langle 9 \rangle_{77} = 9 \\
y^1 &= (\langle 2 \rangle_7, \langle 10 \rangle_{11}) = \langle 0 \rangle_{77} = 0 \\
s_1 &= (\langle 2 \rangle_7, \langle 10 \rangle_{11}) = \langle 65 \rangle_{77} = -12
\end{aligned} \tag{9.28}$$

where the last vector components involve the representation -38, -37, ..., -1, 0, 1, 2, ..., 38 as the elements of Z_{77} . Comparison of Eq. (9.28) and (9.20) shows that the RNS approach indeed provides the correct number.

Caution must be employed because certain moduli can yield singular equations or degenerate inner products as a result of the fact that, in a quotient ring Z_m , $\langle \underline{x} \underline{x} \rangle = 0$ for a non-zero vector \underline{x} . This computation has the disadvantage over the straightforward Gram-Schmidt orthonormalization and Q-R algorithms in that it generates integer growth in the computation at an alarming rate.

9.4 LUT Implementation of Gram-Schmidt Approach

The implementation of the modified Gram-Schmidt procedure described in this section is based solely on the use of residue LUTs and delays. Our objective is to invert matrix A in order to solve a linear system of the form $A\underline{x} = \underline{b}$, by expressing

$$AE = Q \tag{9.29}$$

where E is a triangular matrix and Q has orthogonal columns. Once E and Q are known, we proceed to calculate

$$Q^* Q = A \tag{9.30}$$

and

$$P = Q^{-1} = A^{-1}Q^*. \quad (9.31)$$

We are now ready to invert A by calculating

$$C = A^{-1} = EQ^{-1}. \quad (9.32)$$

The first part of the implementation deals with the calculation of the column vectors of Q, which we note by \underline{w}_j , and the coefficients $a_i \beta_j \gamma_k$, ..., $i = 1, 2, j = i+1, k = j+1, \dots$, which represent the columns (in order) of the matrices E_i . A simple analysis shows that each of the \underline{w}_i can be expressed as

$$\underline{w}_i = \mu_1 \underline{w}_1 + \mu_2 \underline{w}_2 + \mu_3 \underline{w}_3 + \dots + \mu_{i-1} \underline{w}_{i-1} + \mu_i \underline{u}_i \quad (9.33)$$

where μ_k is the k^{th} element of the i^{th} column of the E_i matrix and \underline{u}_i is the i^{th} column of A. To calculate the \underline{w}_i , we need to know the coefficients μ_i , $i = 1, 2, 3, \dots, i-1$, as well as all $\underline{w}_1, \underline{w}_2, \dots, \underline{w}_{i-1}$. This implies that the vector/coefficient calculation is serial and alternates, i.e., we first calculate $\underline{w}_1 = \underline{u}_1$, then we calculate a_1, a_2 followed by \underline{w}_2 then $\beta_1 \beta_2 \beta_3$, etc. The parametric form of the μ_i coefficients can be written as

$$\mu_1 = 1$$

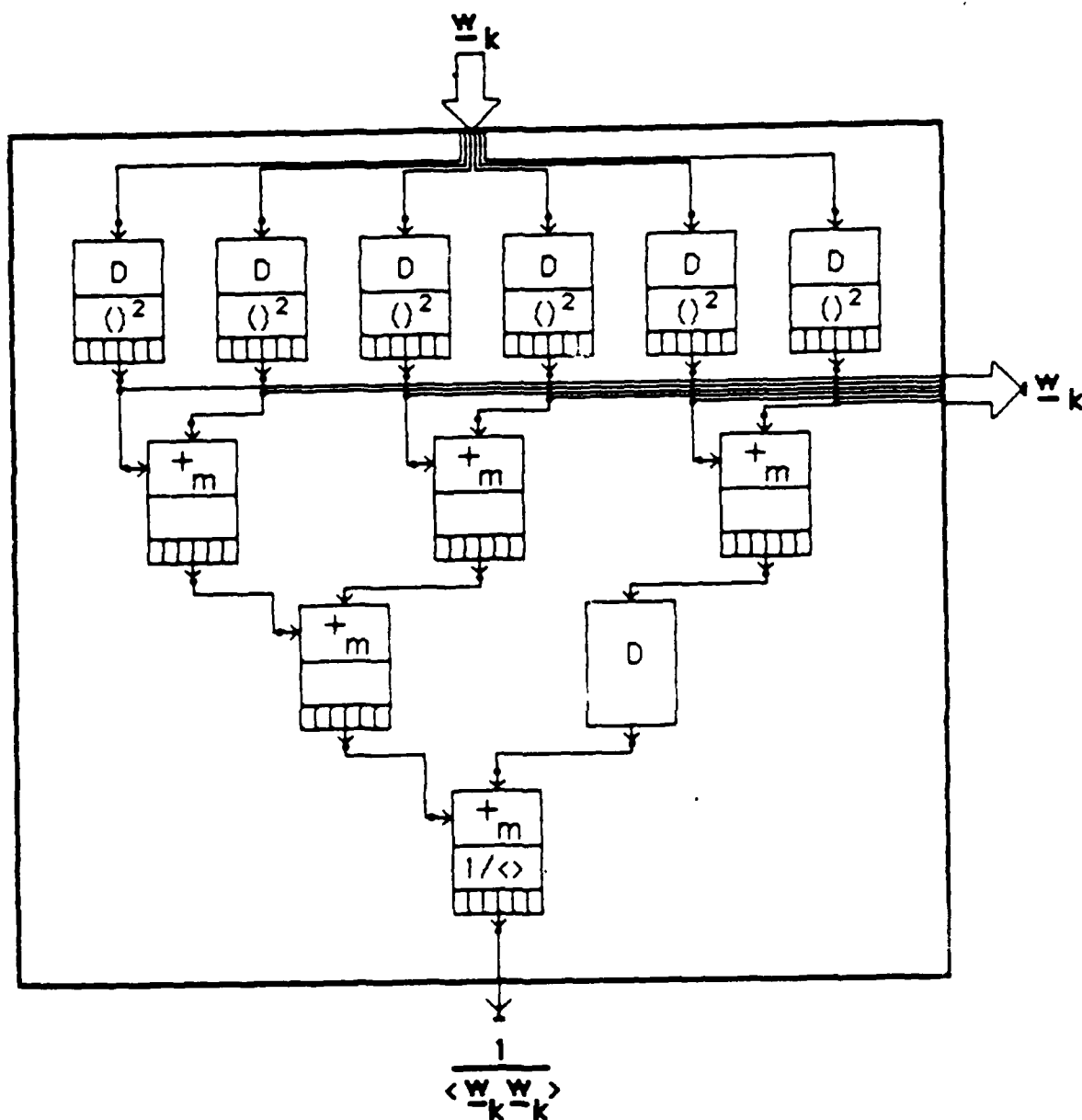
$$\mu_i = \overline{\langle \underline{w}_1 \underline{w}_1 \rangle} / \langle \underline{w}_1 \underline{u}_i \rangle \quad (9.34)$$

$$\mu_2 = \mu_1 \overline{\langle \underline{w}_2 \underline{u}_1 \rangle} / \langle \underline{w}_2 \underline{w}_2 \rangle$$

$$\mu_{i-1} = \mu_1 \overline{\langle \underline{w}_i \underline{u}_i \rangle} / \langle \underline{w}_{i-1} \underline{w}_{i-1} \rangle$$

where $\langle \rangle$ designates the vector dot product and the superscript bar (vinculum) means complement. Eqs. (9.34) reveal that, with the exception of μ_1 and μ_i , the coefficients μ_k have a rather regular form which is a function of μ_i , $(\langle \underline{w}_k \underline{w}_k \rangle)^{-1}$ and $\langle \underline{w}_k \underline{u}_i \rangle$. We thus need a LUT set (LUTS) that is driven by \underline{w}_k and produces an output that is proportional to $(\langle \underline{w}_k \underline{w}_k \rangle)^{-1}$ (LUTS1), and a LUTS that is driven by $\mu_i \underline{u}_i$, \underline{w}_k and $(\langle \underline{w}_k \underline{w}_k \rangle)^{-1}$ and gives two outputs, one that is proportional to μ_k and the other proportional to $\mu_k \underline{w}_k$ (LUTS2). Similar LUTS are needed for the calculation of $\langle \underline{w}_i \underline{w}_i \rangle$ and μ_i (LUTS3 and LUTS4). Figures 9.1 and 9.2 show typical examples of LUTS1 and LUTS2. In these figures each LUT has 2 inputs and 1 output. The top part of the LUT shows the operation performed (multiplication (*) or addition (+)) with respect to modulo m . The middle part of the LUT shows the implementation by wire maps of functions such as dot product complement ($\langle \rangle$) and inverse ($1/\langle \rangle$); when the middle part is blank, no operation is performed there. The lower part symbolizes the output detectors. In the LUTS of Figures 9.1 and 9.2, one can also see blocks that denote delay(s) (denoted by D), which are necessary for data synchronization. To simplify, we designate the equivalent block diagrams of the various LUTS structures and delays with the blocks of Figure 9.3, where the number on the top right corner shows the total delay (in clock cycles) that the LUTS needs in order to provide the lower output, a useful quantity for calculations of delays needed for data synchronization.

With the aid of Figure 9.4, we now describe a pipelined processor that calculates \underline{w}_i and μ_i for a 6x6 example. It can be seen that the processor uses LUTS1 through LUTS4 plus delays and adder LUTs. The inputs of the system are the elements of the matrix A. We assume that all the elements of A are fed into the system in parallel. This is not a necessary condition and is adopted mostly for simplicity. The elements of A are fed through 6 row lines (i.e., a total of 36 lines) which are located at the top left of Figure 9.4. The top row of the system consists of one LUTS4 and 5 LUTS3. The former provides $\langle \underline{w}_1 \underline{w}_1 \rangle$



LUTS1

Figure 9.1 Look-Up Table Set 1 Driven by \underline{w}_k to Produce $1/\langle \underline{w}_k, \underline{w}_k \rangle$
(See text for Explanation of Symbols).

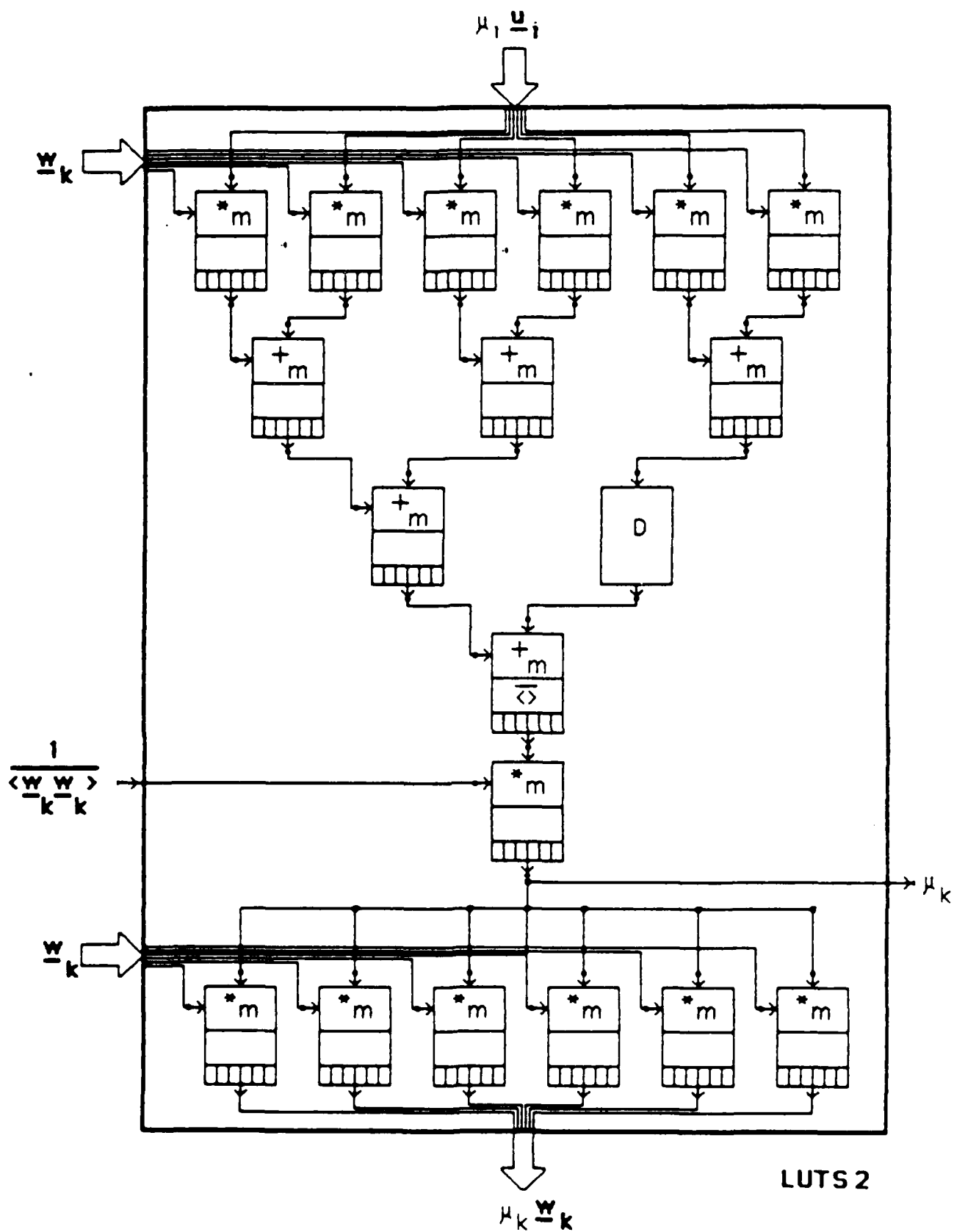
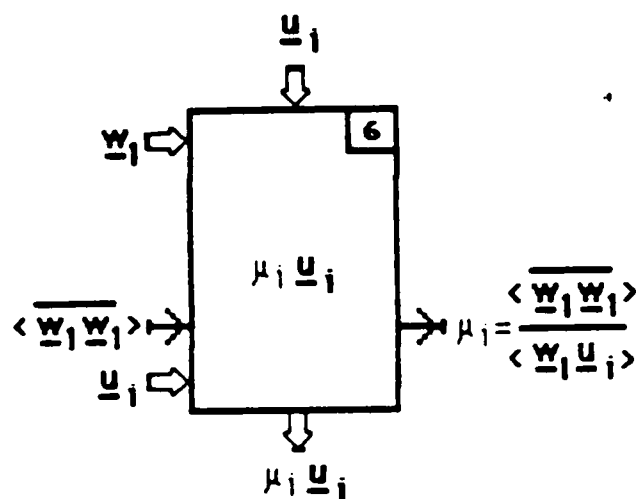
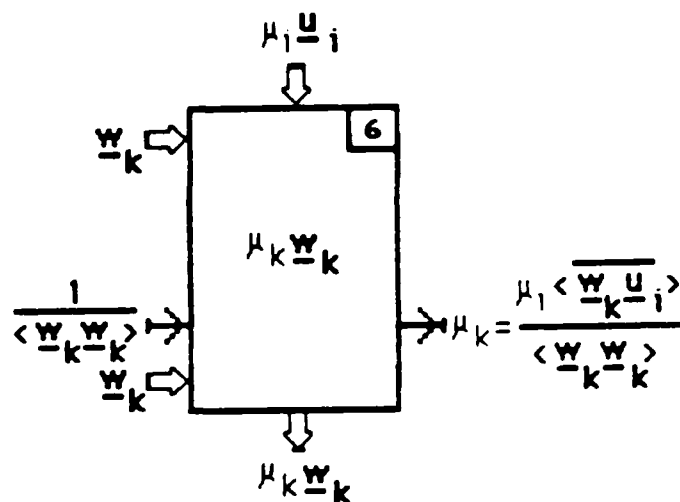


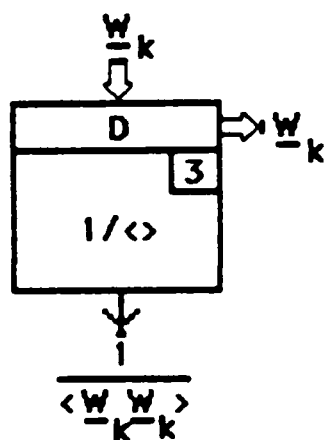
Figure 9.2 Look-Up Table Set 2 Driven by $\mu_1 u_i$, w_k and $1/\langle w_k w_k \rangle$ to Produce μ_k and $\mu_k w_k$.



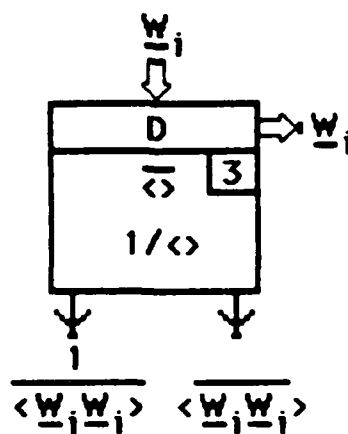
LUTS 4



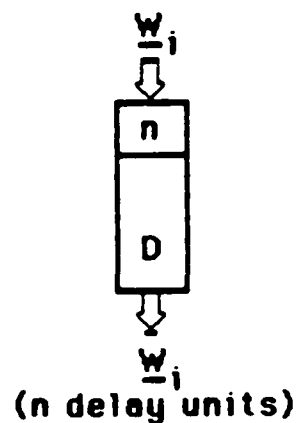
LUTS 2



LUTS 1



LUTS 3



DELAY

Figure 9.3 Equivalent Block Diagrams for Look-Up Table Sets 1 through 4 and Delay Unit for Use in Subsequent Pipeline Processor Schematics.

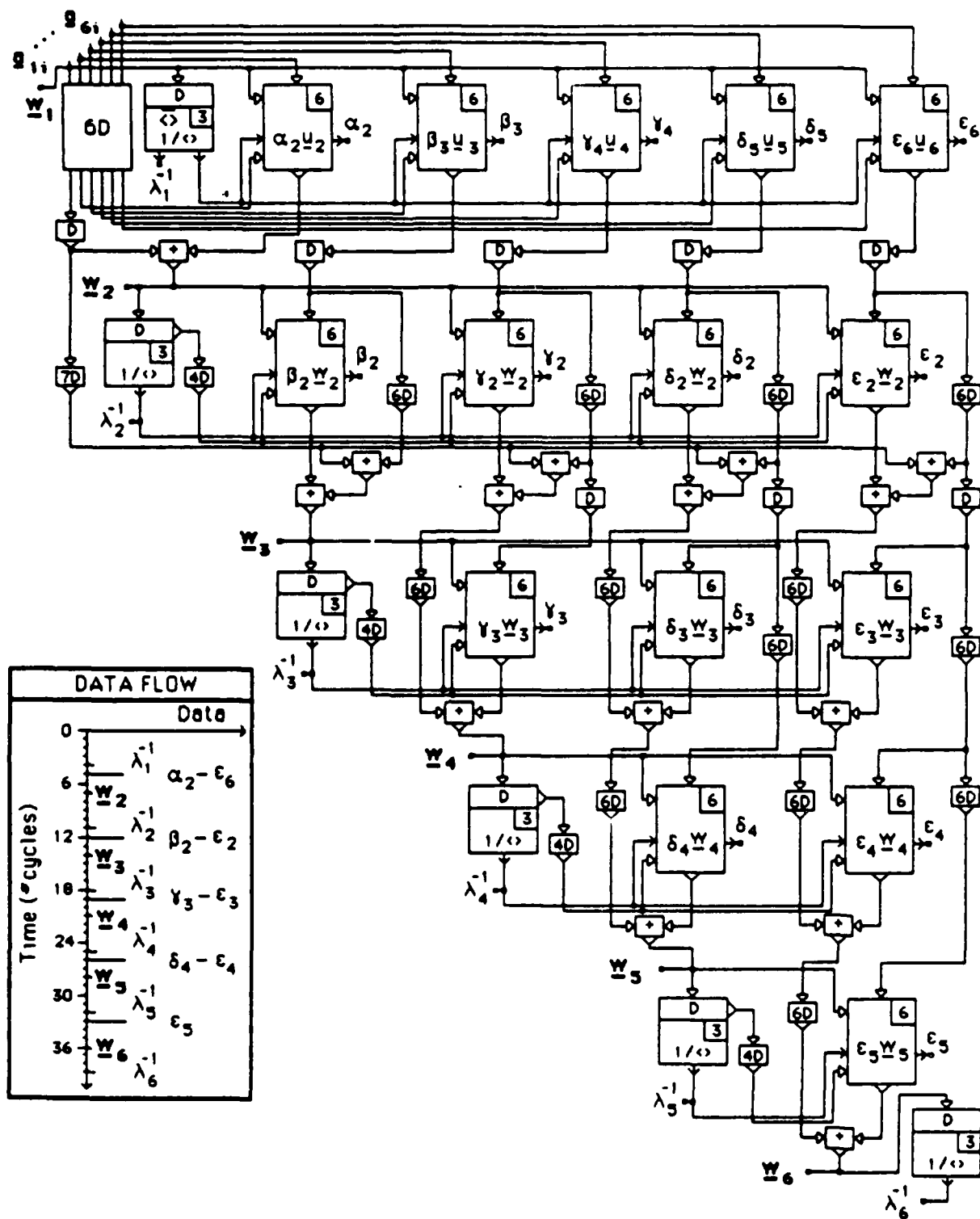


Figure 9.4 Pipelined Processor for Calculating the Elements of Q , E_i and A^{-1} .

which is needed for the calculation of the a_2 through ϵ_8 coefficients. Its output is fed in parallel to the 5 LUTS3 which calculate the above coefficients and the products of the coefficients with the proper u_i 's. The latter are needed for the calculation of the various w_i [see Eq. (9.33)]. For our 6x6 example, it takes 7 cycles for the calculation of w_2 (the 7th cycle being needed for the addition of $a_2 u_2$ and $w_1 = u_1 = a_1$) and 5 cycles for the calculation of $a_2 - \epsilon_8$ coefficients. The w_2 output is connected to a LUTS1 unit while the other outputs (delayed by one clock cycle) are fed into 4 LUTS2 which after 7 cycles calculate w_3 and the coefficients $\gamma_2 - \epsilon_2$. Note that we now need LUT adders in order to add the $\beta_3 u_3$ to $\beta_2 w_2$, the $\gamma_4 u_4$ to $\gamma_2 \epsilon_2$, and so on. This process continues for 4 more rows until all w_i and all the coefficients of the E_i matrices are calculated. It is important to note that the processor of Figure 9.4 operates in a pipelined fashion and thus constantly updates the vectors w_i and the coefficients $a - \epsilon$. This is very important for the APAR scenarios where one wants to constantly update the adaptive weights. Finally, we note that the processor provides output data (vectors and coefficients) every 7 clock cycles (see Figure 9.4).

We now proceed to describe another pipelined processor (Figure 9.5) which is used in order to calculate the elements of the E matrix which in turn are necessary for calculating Eq. (9.32). For our 6x6 example one can easily show that the elements of the triangular E matrix are given by

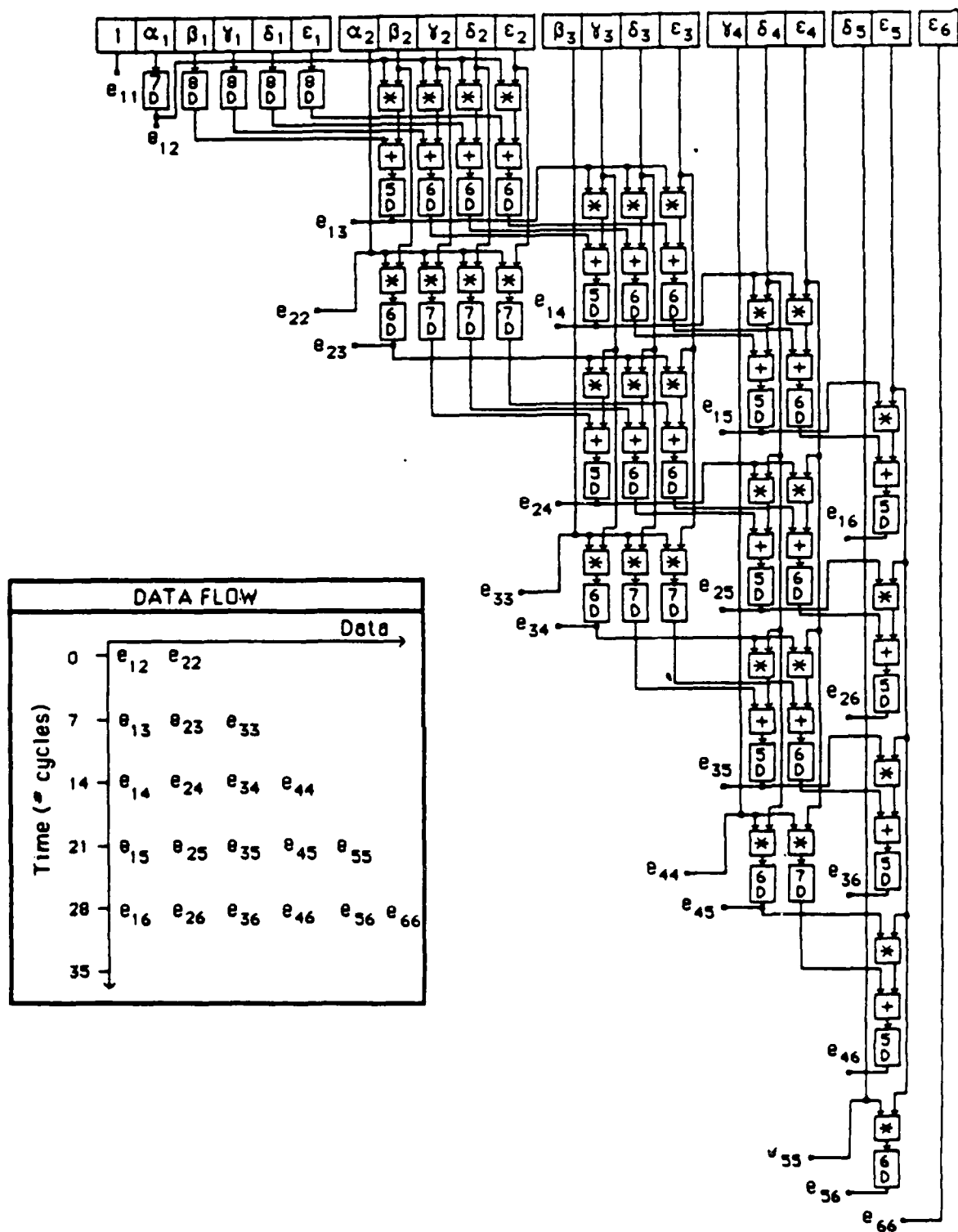


Figure 9.5 Pipelined Processor for Calculating the Products of E_i .

$$e_{11} = 1$$

$$e_{12} = a_1$$

$$e_{13} = \beta_1 + a_1\beta_2$$

$$e_{14} = \gamma_1 + a_1\gamma_2 + e_3\gamma_3$$

$$e_{15} = \delta_1 + a_1\delta_2 + \delta_3\epsilon_{13} + \delta_4e_{14}$$

$$e_{16} = \epsilon_1 + \dots + \epsilon_5e_{15}$$

$$e_{22} = a_2$$

$$e_{23} = \epsilon_2\beta_2$$

$$e_{24} = a_2\gamma_2 + a_2\beta_2\gamma_3$$

$$e_{25} = \dots\dots\dots$$

(9.35)

The form of the elements shows that in calculating the e_{ij} element we need to know all the e_{ij-1} elements as well as all the μ_{i-1} coefficients. This implies that another serial-type operation is necessary. This is exactly what the processor of Figure 9.5 performs; the e_{ij} element is calculated only after the previous e_{im} , $m=1,2,\dots,j-1$, elements have been calculated. Note that in order to avoid unnecessary complexity we have to rearrange the order in which we receive some of the coefficients from the previous processor. Specifically, we have to delay the first set of coefficients by an amount such that we receive all $a_2-\epsilon_2$ at the same clock cycle, all $\beta_3-\epsilon_3$ at the same clock cycle, etc.; thus, we need to delay a_2 by one clock cycle, β_3 by two clock cycles, etc. Once this is done, coefficients with the same subscript will arrive in parallel (see top

of Figure 9.5). It is important to remember that each set of coefficients arrives 7 cycles after the previous set, because this explains the choice of the delays we use. The operations necessary for calculating Eqs. (9.35) can be achieved via the use of multiplier and adder LUTs as well as delays. The $\beta_2-\epsilon_2$ data are fed in parallel into 4 multiplier LUTs which form the products with a_1 . The outputs are then fed into 4 adder LUTs which form the sums with the $\beta_1-\epsilon_1$ data. Note that although all coefficients with subscript 1 are equal to 1, we treat them as unknowns in order to generalize the design of the processor. Since data $\beta_2-\epsilon_2$ are 7 cycles behind the $\beta_1-\epsilon_1$ data, we need to delay the latter by 8 cycles in order to have them available for the addition. The result under the β_2 line (equal to e_{13} , see Eq. (9.35)), is needed for the calculation of the products with the next set of data, which will arrive a total of 5 cycles later. This result (e_{13}) is delayed by 6 cycles and then fed to 3 multipliers which are also driven by data $\gamma_3-\epsilon_3$. These results are then added to the results under lines $\gamma_2-\epsilon_2$ and the e_{14} element is computed. This pipelined process continues until all e_{1i} elements are computed. Note that in parallel to the e_{1i} calculation, we also perform the $e_{2i}-e_{6i}$ calculation (the e_{ji} element does not depend on the $e_{j-1,i}$ element). This is accomplished by driving sets of units similar to the ones we used for the calculation of the e_{1i} elements. Due to the triangular form of the matrix E, the number of units necessary for the calculation of elements $e_{j-1,i}$ is reduced by one as compared with the number of units needed for elements e_{ji} . Due to the pipelining process, the parallel operations and the natural delays of the coefficients, the E matrix elements are computed so that elements with the same second subscript are produced in parallel (see Figure 9.5) and 7 cycles after the previous set. Thus, once again we have achieved the pipelining which is important for high speed processing.

For the evaluation of Eq. (9.32), we also need to calculate the P matrix. One can easily prove that each p_i row vector is given by

$$p_i = \lambda_i^{-1} w_i \quad (9.36)$$

where

$$\lambda_i = 1 / \langle w_i \ w_i \rangle^{-1} \quad (9.37)$$

Eq. (9.37) has already been calculated in the processor of Figure 9.4 because it is needed for the calculation of the α , β , ..., ϵ coefficients. Since w_i appear 4 cycles ahead of λ_i^{-1} , we need to delay them by 4 cycles and subsequently multiply them by λ_i^{-1} . This is shown in Figure 9.6 for all 6 row vectors of the P matrix. We are thus capable of producing all of the elements of a row vector of the P matrix every 7 clock cycles.

This final step in calculating Eq. (9.32) involves a matrix-matrix multiplication; i.e., EP. To achieve this we can use the array processor of Figure 9.7. This system consists of 36 similar units arranged in a square format. Each unit consists of a multiplier and an adder LUT as well as a delay. The LUTs are arranged so that each product is added to the previous one (i.e., we form a multiplier/accumulator). Each set of column units is driven in parallel by the appropriate E data, and each set of row units is driven in parallel by the appropriate P data. Upon summation of 6 products, the adders are read out, and each output is an element of the C matrix. Note that the sequential format required for both E and P data is the same as the sequential format of the data that leave the processors of Figure 9.5 and 9.6. We must multiplex the data, however, because the E and P data come from 21 and 36 output lines, respectively (see Figures 9.5 and 9.6), whereas there are only 6 input lines (per side) for the processor of Figure 9.7. This is not difficult since successive rows (columns) of the E(P) data appear every 7 cycles. Furthermore, by use of appropriate delays, we can provide the

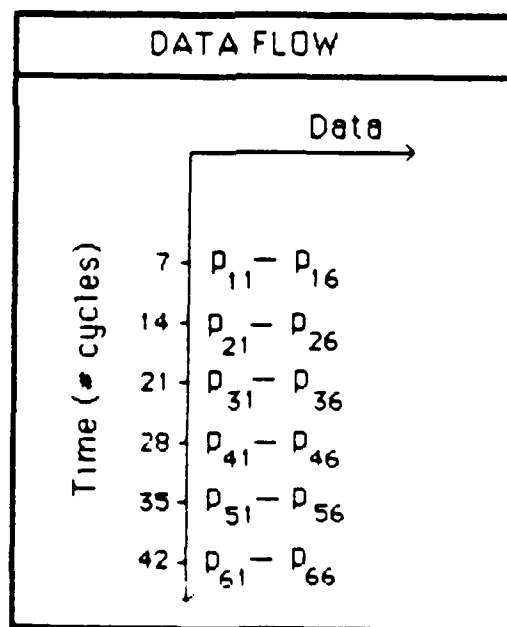
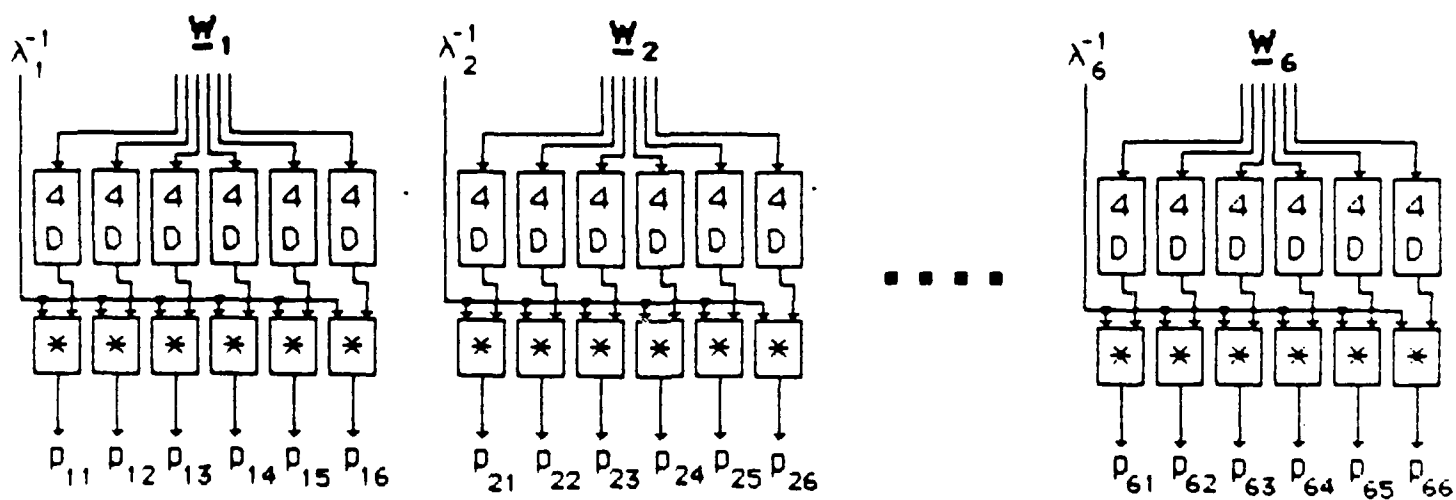


Figure 9.8 Processor for Calculating $P = A^{-1}Q^*$.

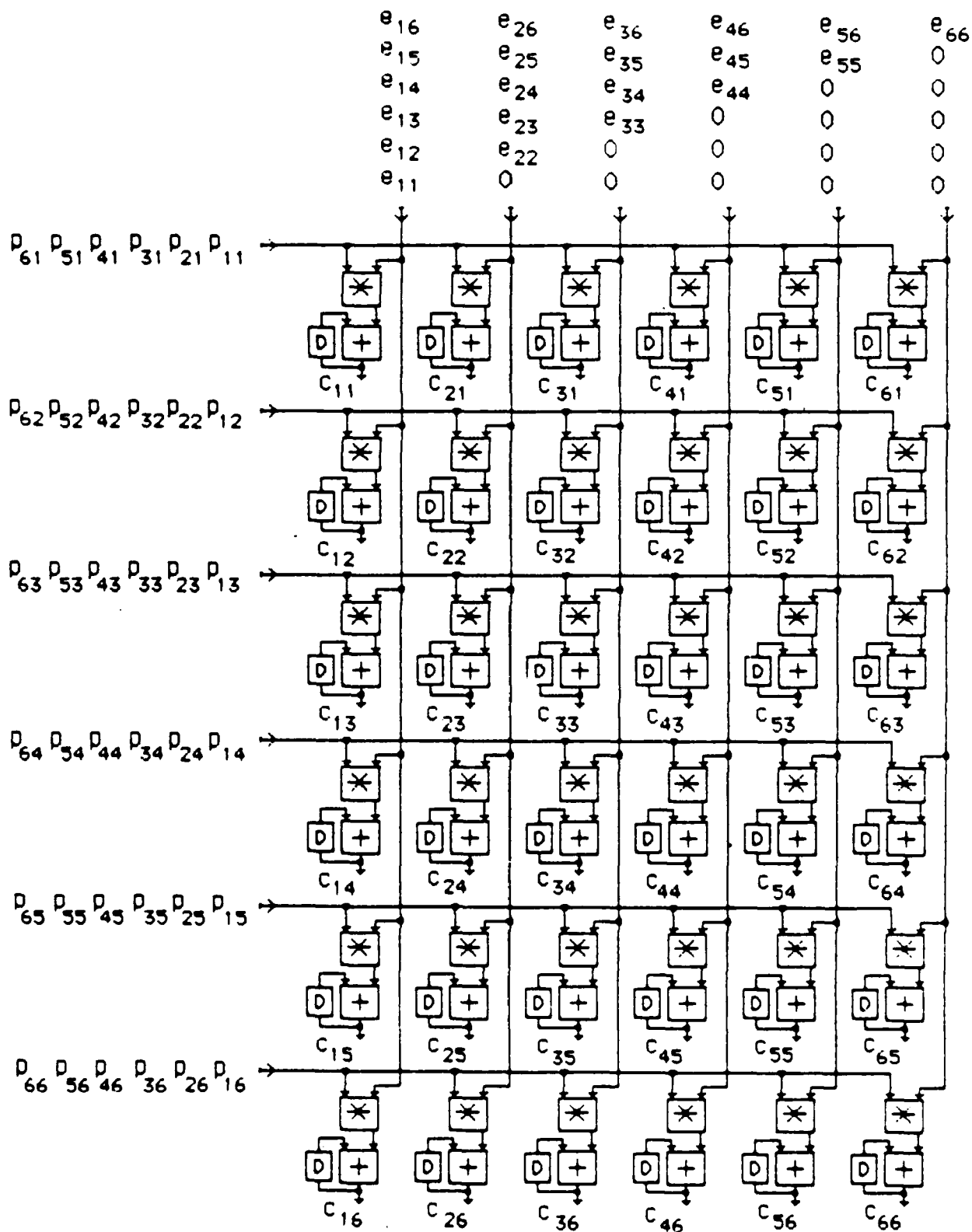


Figure 9.7 Array Processor for Calculating $A^{-1} = EP$.

successive data at one cycle intervals (instead of 7) and allow the array processor to maintain the pipelining for at least 7 cycles, which means that a new matrix can be read out every 7 cycles.

9.5 Processor Characteristics

Let us now discuss the system delay T_g , which is defined as the time required to provide the matrix C after loading matrix A. From Figure 9.4 we see that the calculation of the ϵ_5 coefficients is the most time-consuming operation. It requires a total of

$$t_1 = (n-1) \times (t_{d4} + t_c) \quad (9.38)$$

where n is the dimension of matrix A, and t_{d4} is the delay of LUTS4 and $t_c = D$ is equal to the duration of a clock cycle. The next delay comes from the processor of Figure 9.5 and is proportional to

$$t_2 = t_c + t_{d4} \quad (9.39)$$

We now take into account the delay necessary for interfacing the systems of Figures 9.4 and 9.5. A simple analysis shows that this delay is proportional to

$$t_3 = (n-1) \times t_{d4} \quad (9.40)$$

Finally, the total delay of the array processor of Figure 9.7 is of the order of

$$t_4 = (n+2) \times t_c \quad (9.41)$$

From Eqs. (9.38)-(9.41), we find that the total delay of the system can be approximated by

$$T_s \approx 2n (t_{d4} + t_c) \quad (9.42)$$

To express T_s as a function of t_c we need to calculate the delay t_{d4} . Inspection of Figure 9.2 reveals that the total delay is a function of n . A simple analysis shows that the total delay is proportional to

$$t_{d4} \approx (L + 3) \times t_c \quad (9.43)$$

where L is an integer which satisfies $2^L \leq n < 2^{L+1}$. From Eq. (9.42) and (9.43), we find that the system delay is proportional to

$$T_s \approx 2n (L + 4) t_c \quad (9.44)$$

i.e., the system delay is a linear function of n . Thus for our 6x6 example, the total time required to invert the first matrix is

$$T_s = 72 t_c \quad (9.45)$$

Assuming a clock cycle of the order of 4 nsec, we find that T_s is of the order of 0.4 μ sec. Similarly, for a 12x12 example, T_s is of the order of 0.9 μ sec.

So far we have considered the total delay of the system, i.e., the time to invert the first matrix. Given the pipelined process, however, the second matrix will be inverted after a time T_o which is proportional to

$$T_o \approx nt_c \quad (9.46)$$

This is equal to the time required for the formation of the matrix-matrix multiplication performed by the system of Figure 9.7. In this case and for a 12 x 12 system, each matrix inversion takes about 48 nsec. Note that this dictates the time delay necessary for loading consecutive matrices into the system of Figure 9.4, which is equal to

$$T_1 = nt_c \quad (9.47)$$

We now estimate the total number of LUTs required by the processors of Figures 9.4-9.7. From Figure 9.4 we see that the total number of LUTS1 (or LUTS3) needed is n , whereas the total number of LUTS2 (or LUTS4) is $n(n-1)/2$. The number of LUTs in each LUTS1 (or LUTS3) is about n . Similarly, the number of LUTs in each LUTS2 (or LUTS4) is about $3n$. Finally, we need about $n(n+3)/2$ adder LUTs. Thus, the total number of LUTs in the processor of Figure 9.4 is

$$N_1 = nxn + 3n^2(n-1)/2 + n(n+3)/2 = 3n(n^2+1)/2 \quad (9.48)$$

Similarly, the total number of LUTs for the processor of Figure 9.5 is $\approx n^3/4$, and for the processor of Figure 9.7 is $2n^2$. Thus the total number of LUTs required per modulo is

$$N_t = N_1 + n^3/4 + 2n^2 = (7n^3 + 8n^2 + 6n)/4 \quad (9.49)$$

Note that if m_r is the number of moduli used, then the total number of LUTs needed is $m_r N_t$.

For our 6 x 6 example and with 8-bit input accuracy, we find from Eq (9.5) that M must bound 1.4×10^{13} . To handle this value we use 11

moduli: 7, 9, 11, 13, 17, 19, 23, 25, 29, 31 and 37, and in this case the total number of LUTs becomes $\approx 5,000$. Note that these results reflect the fact that we chose to use a high degree of parallel processing which results in a LUT number requirement that is proportional to n^3 . This requirement can be reduced considerably if we choose to use more of a serial-type processor; this, however, will reduce the speed of the processor. Such issues require trade-off analyses in order to show clearly the optimum system architecture once the convergence time requirement is defined.

10. CONCLUSIONS AND RECOMMENDATIONS

In this program we have examined the possibility of using DMAC-based A0 processors for solving eigensystems in conjunction with the APAR problem. Study of existing eigensystem solution algorithms has revealed that many of the required logical and arithmetic operations cannot be provided by the A0 processors. An analysis of various classes of A0 processors, that are based on DMAC and its parallel extension BPAM, has clearly shown that this type of A0 system offers no advantage over existing all-electronic systems. Therefore, we do not consider this to be a viable approach.

We have suggested that optical interconnections will allow electronic digital multipliers, in square array formats, to be globally interconnected. At high processing speeds (≥ 500 MHz) optical interconnections seem to be the only choice. These, in conjunction with global communications, will enhance the processing speed. We have suggested a simple but efficient fiber-optic technique that allows for global interconnections and we have fabricated a prototype optically addressed digital multiplier. Much work is needed in this area. We suggest that further analyses be carried out of an optically interconnected square array for matrix-matrix multiplication and that a prototype array be built and evaluated.

Residue-based LUT processing has been considered. We have proposed a laser diode-based LUT which can be fabricated with present technology and have fabricated and tested a modulo 7 prototype LUT. The results suggest that when monolithic LUTs are developed, switching speeds that exceed 1 GHz should be easily achievable. We suggest that LUT modelling and analysis take place so that the problem of pulse reflections can be minimized. The number of laser diodes in the fabricated LUT grows as the square of the modulo. In order to avoid an excessive number of

laser diodes novel LUT architectures need to be developed and we recommend that additional research be done in this area. We have shown that B/R and R/B conversions can be efficiently implemented via the use of LUTs. An example of LUT array processing has shown that these conversions require about 20% of the total hardware. This suggests that the longer the processing in the RNS the less the relative hardware needed for conversions. For the above example we have also shown that the total number of gates for the RNS LUT processing is about twice that of the electronic counterpart. This suggests that comparison analyses be done in order to identify both the competitiveness of LUT processing as well as the applications for which RNS LUT processing is well suited.

The RNS LUT processing has also been studied for use in the APAR area. We have found that the only algorithm suited for residue LUT implementation is a variant of the Gram-Schmidt orthogonalization procedure. We have shown, through examples, that such an approach yields the correct results. We recommend that this approach be further analysed in order to determine its exact requirements and shortcomings. Finally, we have presented the complete design of a pipelined RNS LUT processor for the inversion of a 6×6 APAR data matrix. We have shown that with a fully parallel implementation, the matrix inversion takes place in $N+1$ cycles. Note that for such an implementation, the total number of LUTs grows as $\approx 2N^3$. Thus, in order to avoid an excessive number of LUTs we recommend that a similar design be made with a more serial nature. Such a design should clearly show the trade-offs between processing speed and hardware complexity.

11. ACKNOWLEDGEMENT

The contributions to this report by Dr. P. N. Tamura and Mr. J. Zomp of the Westinghouse R&D Center, Pittsburgh, PA, Drs. E. C. Malarkey, J. Bradley and P. Beaudet of Westinghouse ATD, Baltimore, MD, and Dr. B.V.K.V. Kumar of Carnegie-Mellon University is greatly acknowledged. We also thank Mr. L. D. Kurtz for his technical support in the experimental work.

This report was typed by Toni McElhaney and Marilyn Cross.

REFERENCES

1. G. H. Golub and C. F. Van Loan, "Matrix Computations," John Hopkins University Press, Baltimore (1983).
2. J. H. Wilkinson, "The Algebraic Eigenvalue Problem," Oxford University Press, Oxford (UK), (1965).
3. H. J. Whitehouse and J. M. Speiser, "Aspects of Signal Processing, Part 2." G. Tacconi, Ed., pp. 669-702, Proc. NATO Advanced Study Institute, D. Reidel Publishing Company, Boston (1978).
4. R. P. Bocker, S. R. Clayton, and K. Bromley, "Electrooptical Matrix Multiplication Using the Twos Complement Arithmetic for Improved Accuracy," Appl. Optics, 22, 2019 (1983).
5. A. P. Goutsoulis, "On the System Efficiency of Digital Accuracy A0 Processors," presented at the SPIE 1986 Southeast Conference, Orlando, FL, April 1986.
6. D. Psaltis, private communication.
7. P. S. Guilfoyle, "Systolic Acousto-Optic Binary Convolver," Opt. Eng. 23, 020 (1984).
8. M. Carlotto and D. Casasent, "Microprocessor-Based Fiber-Optic Iterative Optical Processor," Appl. Optics, 21, 147 (1982).
9. A. Wilson, "ISSCC Highlights IC Advances," Elect. Imaging, 3, 56 (1984).
10. D. Maydan, "Acousto-Optical Pulse Modulators," IEEE J. of Quantum Effects, QE-6 15 (1970).
11. F. S. Lee et al., "A High-Speed LSI GaAs 8x8 Bit Parallel Multiplier," IEEE J. of Sol. St. Cir., SC-17, 638 (1982).
12. I. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, 34, 349 (1965).
13. J. W. Goodman et al., "Optical Interconnections for VLSI Systems," Proc. of IEEE, 72, 850 (1984).
14. R. K. Jain and D. E. Snyder, "Switching Characteristics at Logic Gates Addressed by Picosecond Light Pulses," IEEE J. Quantum Electr., QE-19, 558 (1983).

15. S. Y. Kung, "On Supercomputing with Systolic/Wavefront Array Processors," *Proc. IEEE*, 72, 867 (1984).
16. "MECL System Design Handbook," Motorola, May 1980.
17. K. Hwang, "Computer Arithmetic: Principles, Architecture and Design," J. Wiley and Sons, New York (1979).
18. S. D. Pesaris, "A 40 nsec 17x17 Bit Array Multiplier," *IEEE Trans. Computers*, C-20, 442 (1971).
19. N. S. Szabo and R. I. Tanaka, "Residue Arithmetic and Its Applications to Computer Technology," McGraw-Hill, New York, 1967.
20. A. Huang, Y. Tsunoda, J. W. Goodman, and S. Ishihara, "Optical Computation Using Residue Arithmetic," *Appl. Optics*, 18, 149 (1979).
21. A. Tai, I. Cindrich, J. R. Fienup, and C. C. Aleksoff, "Optical Residue Arithmetic Computer with Programmable Computation Modules," *Applied Optics*, 18, 2812 (1979).
22. J. N. Polky, D. D. Miller, and R. L. Gutman, "Optical Residue Arithmetic Data Processing," Final Report, Contract No. F33615-80-C-110, Report. No. AFWAL-TR-81-1244, February 1982.
23. T. K. Gaylord, M. M. Mirsalehi, and C. C. Guest, "Optical Digital Truth Table Look-Up Processing," *Opt. Eng.* 24, 048 (1985).
24. R. S. Tucker, "High-Speed Modulation of Semiconductor Lasers," Invited Paper, *J. Lightwave Technol.* LT-3, 1180 (1985) and references therein.
25. E. C. Goodfellow et al., "Optoelectronic Components for Multigigabit Systems," Invited Paper, *J. Lightwave Technol.* LT-3, 1170 (1985) and references 10 to 12 therein.
26. T. H. Wood, C. A. Burrus, D. A. B. Miller, D. S. Chemla, T. C. Damen, A. C. Gossard, and W. Wiegmann, "High-Speed Optical Modulation with GaAs/GaAlAs Quantum Wells in a p-i-n Diode Structure," *Appl. Phys. Lett.* 44, 16 (1984).
27. M. Newman, "Solving Equations Exactly," *N.B.S. Jour. of Res.*, 71B, 171 (1967).

APPENDIX A AN ANALYSIS FOR CIRCULARLY POLARIZED SAMPLING

This appendix describes the mathematical validity of the circularly polarized sampling by observing the sampled function in the frequency domain.

Consider a complex function, $f(t)$, that has a real and an imaginary part:

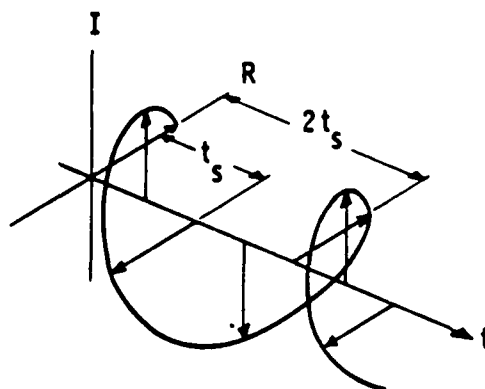
$$f(t) = f_R(t) + f_I(t) \quad (A-1)$$

The sampling function is a series of delta functions, the phase of which is shifted by 90° (Figure A.1) or the complex amplitude part forms a series $(1, j, -1, -j, 1, j, \dots)$. In the first sampling period, it samples the real part of the input function; in the second sampling period, it samples the imaginary part; in the third period, it samples the real part with negative polarity; and in the fourth period, it samples the imaginary part with negative polarity. This four-cycle pattern is repeated for the remainder of the sampling operation. We call this function a Right-Hand Circularly Polarized (RCP) sampling function, indicating the rotational orientation of the phasor. Similarly, a Left-Hand Circularly Polarized (LCP) sampling function is a delta function series with the quadrature rotating in the opposite direction $(1, -j, -1, j, \dots)$ and the sampling operation can be performed in a similar way.

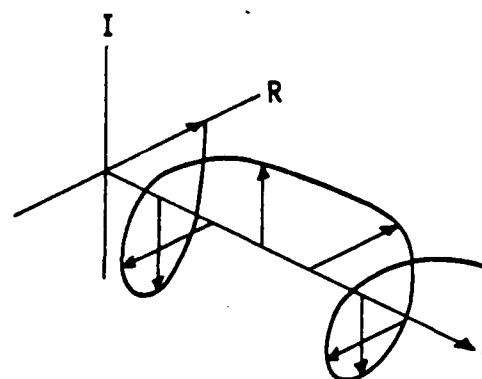
Mathematically this operation can be described as follows:

Let $s_{RCP}(t)$ and $s_{LCP}(t)$ be the RCP and LCP sampling functions, respectively, i.e.,

$$s_{RCP}(t) = e^{j2\pi \frac{t}{2t_s}} \text{comb}\left(\frac{2t}{t_s}\right) \quad (A-2)$$



a) Right Hand Circularly Polarized
Sampling Function
(1, j, -1, -j, ...)



b) Left Hand Circularly Polarized
Sampling Function
(1, -j, -1, j, ...)

Figure A.1 Circularly polarized sampling function.

$$s_{LCP}(t) = e^{-j2\pi \frac{t}{2t_s}} \text{comb} \left(\frac{2t}{t_s} \right) \quad (\text{A-3})$$

The sampled versions of the functions $f_{RCP}(t)$ and $f_{LCP}(t)$ are obtained by evaluating the real part of the product of $f(t)$ and the sampling functions

$$\begin{aligned} f_{RCP}(t) &= \text{Re} \left\{ f(t) s_{RCP}(t) \right\} \\ &= \text{Re} \left\{ \left[f_R(t) + j f_I(t) \right] \cdot \left[\cos 2\pi \frac{t}{2t_s} + j \sin 2\pi \frac{t}{2t_s} \right] \text{comb} \left(\frac{2t}{t_s} \right) \right\} \\ &= \frac{1}{2} \left[f(t) e^{j2\pi \frac{t}{2t_s}} + f^*(t) e^{-j2\pi \frac{t}{2t_s}} \right] \text{comb} \left(\frac{2t}{t_s} \right) \quad (\text{A-4}) \end{aligned}$$

$$\begin{aligned}
f_{\text{LCP}}(t) &= \text{Re} \left\{ f(t) s_{\text{LCP}}(t) \right\} \\
&= \frac{1}{2} \left[f^*(t) e^{j2\pi \frac{t}{2t_s}} + f(t) e^{-j2\pi \frac{t}{2t_s}} \right] \text{comb} \left(\frac{t}{t_s} \right) \quad (\text{A-5})
\end{aligned}$$

Therefore $f_{\text{RCP}}(t) = f_{\text{LCP}}^*(t)$, i.e., the RCP sampled signal and the LCP sampled signal are conjugate to each other.

The spectra of these signals can be obtained by Fourier transformation of the expressions:

$$F_{\text{RCP}}(\nu) = \left[F\left(\nu - \frac{1}{2t_s}\right) + F^*\left(-\nu + \frac{1}{2t_s}\right) \right] * \text{comb} \left(\frac{t_s}{2} \nu \right) \quad (\text{A-6})$$

$$F_{\text{LCP}}(\nu) = \left[F^*\left(-\nu - \frac{1}{2t_s}\right) + F\left(\nu + \frac{1}{2t_s}\right) \right] * \text{comb} \left(\frac{t_s}{2} \nu \right), \quad (\text{A-7})$$

after dropping unnecessary coefficients (see Figure A.2). These equations indicate that the frequency domain contains the replications of the original spectra as in the case of ordinary sampling. The differences are that the primary spectra are located at $1/2t_s$ and that the adjacent aliases are conjugate to each other. The spacing between the alias centers is $1/t_s$. Therefore, the bandwidth of the original function B must be smaller than that of the sampling frequency, i.e.,

$$B < \frac{1}{t_s} \quad (\text{A-8})$$

In other words, the bandwidth requirement is identical to that of conventional sampling, namely, the Nyquist criterion.

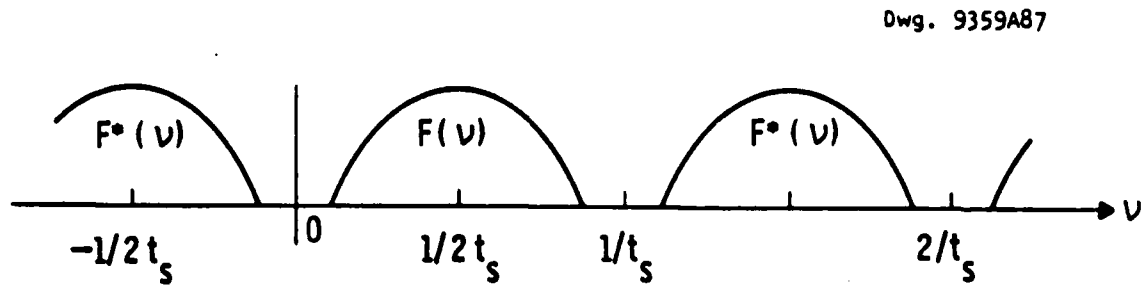


Figure A.2 The spectrum for a CP sampled signal.

END

8-87

DTIC